OpenVMS DCL Dictionary: A–M

December 1995

This manual provides detailed reference information and examples for OpenVMS DCL commands and lexical functions.

Revision/Update Information: This manual supersedes the *OpenVMS DCL Dictionary*, *OpenVMS Version 6.2*.

Software Version: OpenVMS Alpha Version 7.0
OpenVMS VAX Version 7.0

Digital Equipment Corporation
Maynard, Massachusetts

December 1995

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from Digital or an authorized sublicensor.

Digital conducts its business in a manner that conserves the environment and protects the safety and health of its employees, customers, and the community.

© Digital Equipment Corporation 1995. All rights reserved.

The following are trademarks of Digital Equipment Corporation: AXP, Bookreader, CDA, CI, DEC, DECdirect, DECdtm, DECevent, DECnet, DECprint, DECram, DECspell, DECterm, DECUS, DECwindows, DECwrite, DECwriter Correspondent, DEQNA, DTIF, EDT, GIGI, HSC, LA, LiveLink, LN03, MASSBUS, MicroVAX, MSCP, OpenVMS, PrintServer 40, Q–bus, ReGIS, RQDX3, RX02, RX33, RZ, TMSCP, ULTRIX, UNIBUS, VAX, VAXcluster, VAX COBOL, VAX FORTRAN, VAXft, VAXserver, VAXstation, VMS, VMScluster, VT, VT100, VT300, WPS, XUI, and the DIGITAL logo.

The following are third-party trademarks:

Adobe, Display PostScript, and PostScript are registered trademarks of Adobe Systems Incorporated.

OSF, OSF/Motif, and Motif are registered trademarks and Open Software Foundation is a trademark of the Open Software Foundation, Inc.

UNIX is a registered trademark licensed exclusively by X/Open Company, Ltd.

All other trademarks and registered trademarks are the property of their respective holders.

ZK6199

This document is available on CD–ROM.

Contents

Preface vii

= (Assignment Statement) DCLI–1
:= (String Assignment) DCLI–5
@ (Execute Procedure) DCLI–9
ACCOUNTING DCLI–14
ALLOCATE DCLI–15
ANALYZE/AUDIT DCLI–18
ANALYZE/CRASH_DUMP DCLI–19
ANALYZE/DISK_STRUCTURE DCLI–20
ANALYZE/ERROR_LOG DCLI–21
ANALYZE/IMAGE DCLI–22
ANALYZE/MEDIA DCLI–25
ANALYZE/OBJECT DCLI–26
ANALYZE/PROCESS_DUMP DCLI–30
ANALYZE/RMS_FILE DCLI–34
ANALYZE/SYSTEM DCLI–35
APPEND DCLI–36
ASSIGN DCLI–41
ASSIGN/MERGE DCLI–47
ASSIGN/QUEUE DCLI–48
ATTACH DCLI–50
BACKUP DCLI–52
CALL DCLI–53
CANCEL DCLI–57
CLOSE DCLI–59
CONNECT DCLI–61
CONTINUE DCLI–64
CONVERT DCLI–65
CONVERT/DOCUMENT DCLI–66
CONVERT/RECLAIM DCLI–78
COPY DCLI–79
COPY/FTP DCLI–89
COPY/RCP DCLI–91
CREATE DCLI–93
CREATE/DIRECTORY DCLI–96
CREATE/FDL DCLI–99
CREATE/NAME_TABLE DCLI–100
CREATE/TERMINAL DCLI–104

DEALLOCATE DCLI–110
DEASSIGN DCLI–111
DEASSIGN/QUEUE DCLI–115
DEBUG DCLI–116
DECK DCLI–119
DEFINE DCLI–122
DEFINE/CHARACTERISTIC DCLI–128
DEFINE/FORM DCLI–130
DEFINE/KEY DCLI–134
DELETE DCLI–138
DELETE/CHARACTERISTIC DCLI–142
DELETE/ENTRY DCLI–143
DELETE/FORM DCLI–146
DELETE/INTRUSION_RECORD DCLI–147
DELETE/KEY DCLI–148
DELETE/QUEUE DCLI–150
DELETE/QUEUE/MANAGER DCLI–152
DELETE/SYMBOL DCLI–153
DEPOSIT DCLI–155
DIAGNOSE (Alpha Only) DCLI–159
DIFFERENCES DCLI–160
DIRECTORY DCLI–170
DISABLE AUTOSTART DCLI–182
DISCONNECT DCLI–184
DISMOUNT DCLI–186
DUMP DCLI–190
EDIT/ACL DCLI–199
EDIT/EDT DCLI–200
EDIT/FDL DCLI–204
EDIT/SUM DCLI–205
EDIT/TECO DCLI–206
EDIT/TPU DCLI–209
ENABLE AUTOSTART DCLI–210
ENDSUBROUTINE DCLI–213
EOD DCLI–214
EOJ DCLI–216
EXAMINE DCLI–217
EXCHANGE DCLI–220
EXCHANGE/NETWORK DCLI–221
EXIT DCLI–230
FONT DCLI–234
GOSUB DCLI–235
GOTO DCLI–237
HELP DCLI–239
HELP/MESSAGE DCLI–247
IF DCLI–253
INITIALIZE DCLI–256

INITIALIZE/QUEUE DCLI–266
INQUIRE DCLI–281
INSTALL DCLI–284
JOB DCLI–285

Lexical Functions DCLI–291

F$CONTEXT DCLI–294
F$CSID DCLI–300
F$CVSI DCLI–302
F$CVTIME DCLI–304
F$CVUI DCLI–306
F$DEVICE DCLI–307
F$DIRECTORY DCLI–309
F$EDIT DCLI–310
F$ELEMENT DCLI–312
F$ENVIRONMENT DCLI–314
F$EXTRACT DCLI–317
F$FAO DCLI–319
F$FILE_ATTRIBUTES DCLI–326
F$GETDVI DCLI–329
F$GETJPI DCLI–345
F$GETQUI DCLI–351
F$GETSYI DCLI–374
F$IDENTIFIER DCLI–380
F$INTEGER DCLI–382
F$LENGTH DCLI–383
F$LOCATE DCLI–384
F$MESSAGE DCLI–386
F$MODE DCLI–388
F$PARSE DCLI–390
F$PID DCLI–393
F$PRIVILEGE DCLI–395
F$PROCESS DCLI–396
F$SEARCH DCLI–397
F$SETPRV DCLI–400
F$STRING DCLI–404
F$TIME DCLI–405
F$TRNLNM DCLI–406
F$TYPE DCLI–411
F$USER DCLI–413
F$VERIFY DCLI–414

LIBRARY DCLI–416
LICENSE DCLI–417
LINK DCLI–418
LOGIN Procedure DCLI–419
LOGOUT DCLI–423
MACRO DCLI–424

MAIL DCLI–431
MERGE DCLI–432
MESSAGE DCLI–433
MONITOR DCLI–434
MOUNT DCLI–435

Index

Tables

DCLI–1 CPU Time Limit Specifications and Actions DCLI–272
DCLI–2 Working Set Default, Extent, and Quota Decision DCLI–279
DCLI–3 Summary of Lexical Functions DCLI–291
DCLI–4 Summary of FAO Directives DCLI–321
DCLI–5 F$FILE_ATTRIBUTES Items DCLI–326
DCLI–6 F$GETDVI Items DCLI–330
DCLI–7 Values Returned by the DEVCLASS Item DCLI–337
DCLI–8 Values Returned by the DEVTYPE Item DCLI–338
DCLI–9 F$GETJPI Items DCLI–346
DCLI–10 F$GETQUI Keywords DCLI–353
DCLI–11 F$GETQUI Items DCLI–355
DCLI–12 F$GETSYI Items for the Local Node Only DCLI–375
DCLI–13 F$GETSYI Items for the Local Node or for Other Nodes in the VAXcluster DCLI–378
DCLI–14 F$MESSAGE Keywords DCLI–386
DCLI–15 Context Symbol Types DCLI–411

Preface

Intended Audience

This manual is intended for all users of the OpenVMS operating system. It includes descriptions of all DIGITAL Command Language (DCL) commands and lexical functions. If a command has any restrictions or requires special privileges, they are noted in reference information for that command.

Readers of this manual should be familiar with the material covered in the *OpenVMS User's Manual*.

Document Structure

This manual contains detailed descriptions of each command and lexical function. The commands are listed in alphabetical order, with the command name appearing at the top of every page. The lexical functions are grouped under the heading Lexical Functions (after the JOB command description) and are listed alphabetically within that grouping; the lexical function name appears at the top of each page.

The hard copy of the *OpenVMS DCL Dictionary* is a two-part manual. The first part contains commands beginning with the letters A to M (including the lexical functions). The second part contains commands beginning with the letters N to Z. The Table of Contents and Index are comprehensive: they include both parts. The Bookreader copy of the *OpenVMS DCL Dictionary* is one manual.

Appendix A lists the obsolete DCL commands and the current services that replace them.

The commands that invoke language compilers and other OpenVMS optional software products are not included in this manual; they are included in the documentation provided with those products.

Related Documents

For an introduction to the OpenVMS operating system and for information on using DCL, see the *OpenVMS User's Manual*. This manual is especially recommended for novice users or users lacking experience with interactive computer systems.

The *OpenVMS User's Manual* provides an overview of DCL command language concepts and defines and illustrates good practices in constructing command procedures with DCL commands and lexical functions.

See the various utilities reference manuals for detailed information on utilities. These manuals describe the DCL commands that invoke the various utilities, describe any commands that you can enter while running a utility, and provide reference information. The *OpenVMS DCL Dictionary* only provides a brief description and format information of each utility.

For message descriptions, use the online Help Message utility or refer to the OpenVMS system messages documentation.

Reader's Comments

Digital welcomes your comments on this manual.

Print or edit the online form SYS$HELP:OPENVMSDOC_COMMENTS.TXT and send us your comments by:

Internet	**openvmsdoc@zko.mts.dec.com**
Fax	603 881-0120, Attention: OpenVMS Documentation, ZK03-4/U08
Mail	OpenVMS Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How To Order Additional Documentation

Use the following table to order additional documentation or information. If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Location	Call	Fax	Write
U.S.A.	DECdirect 800–DIGITAL 800–344–4825	Fax: 800–234–2298	Digital Equipment Corporation P.O. Box CS2008 Nashua, NH 03061
Puerto Rico	809–781–0505	Fax: 809–749–8300	Digital Equipment Caribbean, Inc. 3 Digital Plaza, 1st Street, Suite 200 P.O. Box 11038 Metro Office Park San Juan, Puerto Rico 00910–2138
Canada	800–267–6215	Fax: 613–592–1946	Digital Equipment of Canada, Ltd. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: DECdirect Sales
International	—	—	Local Digital subsidiary or approved distributor
Internal Orders	DTN: 264–4446 603–884–4446	Fax: 603–884–3960	U.S. Software Supply Business Digital Equipment Corporation 10 Cotton Road Nashua, NH 03063–1260

ZK–7654A–GE

Conventions

The name of the OpenVMS AXP operating system has been changed to OpenVMS Alpha. Any references to OpenVMS AXP or AXP are synonymous with OpenVMS Alpha or Alpha.

The following conventions are used to identify information specific to OpenVMS Alpha or to OpenVMS VAX:

Alpha	The Alpha icon denotes the beginning of information specific to OpenVMS Alpha.
VAX	The VAX icon denotes the beginning of information specific to OpenVMS VAX.
◆	The diamond symbol denotes the end of a section of information specific to OpenVMS Alpha or to OpenVMS VAX.

In this manual, every use of DECwindows and DECwindows Motif refers to DECwindows Motif for OpenVMS software.

The following conventions are also used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 (GOLD) *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 (defined as the GOLD key), then press and release another key or a pointing device button. The GOLD key definition is often mapped to the PF1 key on the keypad. GOLD key sequences can also have a slash (/), dash (–), or underscore (_) as a delimiter in EVE commands.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
. . .	Horizontal ellipsis points in examples indicate one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	Vertical ellipsis points indicate the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.

[ ]	In command format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in an OpenVMS file specification or in the syntax of a substring specification in an assignment statement.)
{ }	In command format descriptions, braces indicate a required choice of options; you must choose one of the options listed.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in Bookreader versions of the manual.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Italic text also represents information that can vary in system messages (for example, Internal error *number*), command lines (for example, /PRODUCER=*name*), and command parameters in text.
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
–	A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

The contents of the display examples for certain commands described in this manual may differ slightly from the actual output provided by these commands on your system. However, when the behavior of a command differs significantly between Alpha and VAX, that behavior is described in text and rendered, as appropriate, in separate examples.

= (Assignment Statement)

Defines a symbolic name for a character string or integer value.

Format

symbol-name =[=] expression

symbol-name[bit-position,size] =[=] replacement-expression

Note

Digital advises against assigning a symbolic name that is already a DCL command name. Digital especially discourages the assignment of symbols such as IF, THEN, ELSE, and GOTO, which can affect the interpretation of command procedures.

Parameters

symbol-name

Specifies a string of 1 to 255 characters for the symbol name. The name can contain any alphanumeric characters from the DEC Multinational character set, the underscore (_), and the dollar sign ($). However, the name must begin *only* with an alphabetic character (uppercase and lowercase characters are equivalent), an underscore, or a dollar sign. Using one equal sign (=) places the symbol name in the local symbol table for the current command level. Using two equal signs (==) places the symbol name in the global symbol table.

expression

Names the value on the right-hand side of an assignment statement. This parameter can consist of a character string, an integer, a symbol name, a lexical function, or a combination of these entities. The components of the expression are evaluated, and the result is assigned to the symbol. All literal character strings must be enclosed in quotation marks (‘‘ ’’). If the expression contains a symbol, the expression is evaluated using the symbol’s value.

The result of expression evaluation is either a character string or a signed integer value. If the expression is evaluated as a string, the symbol is assigned a string value. If the expression is evaluated as an integer, the symbol is assigned an integer value. If the integer value exceeds the capacity of the 4-byte buffer that holds it, no error message is issued.

For a summary of operators used in expressions, details on how to specify expressions, and details on how expressions are evaluated, see the *OpenVMS User’s Manual*.

DCL uses a buffer that is 1024 bytes long to hold an assignment statement and to evaluate the expression. The length of the symbol name, the expression, and the expression’s calculations cannot exceed 1024 bytes.

[bit-position,size]

States that a binary overlay is to be inserted in the current 32-bit value of a symbol name. The current value of the symbol name is evaluated. Then, the specified number of bits is replaced by the result of the replacement expression. The bit position is the location relative to bit 0 at which the overlay is to occur.

If the symbol you are overlaying is an integer, then the bit position must be less than 32. The sum of the bit position and the size must be less than or equal to 32.

If the symbol you are overlaying is a string, then the bit position must be less than 6152. Because each character is represented using 8 bits, you can begin an overlay at any character through the 768th character. (The 768th character starts in bit position 6144.) The sum of the bit position and the size must be less than or equal to 6152.

The size is the number of bits to be overlaid. If you specify a size that is greater than 32, DCL reduces the size to 32.

The brackets are required notation; no spaces are allowed between the symbol name and the left bracket. Specify values for the bit position and size as integers.

replacement-expression

Specifies the value that is used to overlay the symbol you are modifying. Specify the replacement expression as an integer.

If the symbol you are modifying is an integer, the replacement expression defines a bit pattern that is overlaid on the value assigned to the symbol. If the symbol you are modifying is a character string, the result of the replacement expression defines a bit pattern that is overlaid on the specified bits of the character string. If the symbol you are modifying is undefined, the result of the replacement expression is overlaid on a null string.

Description

Symbols defined using assignment statements allow you to extend the command language. At the interactive command level, you can use symbols to define synonyms for commands or command lines. In command procedure files, you can use symbols to provide for conditional execution and substitution of variables.

The maximum number of symbols that can be defined at any time depends on the following:

- The amount of space available to the command interpreter to contain symbol tables and labels for the current process. The amount of space is determined for each process by the system parameter CLISYMTBL.
- The size of the symbol names and their values. The command interpreter allocates space for a symbol name and its value. In addition, a few bytes of overhead are allocated for each symbol.

Examples

1.
```
$ LIST == "DIRECTORY"
```

   The assignment statement in this example assigns the user-defined synonym LIST as a global symbol definition for the DCL command DIRECTORY.

2.
```
$ COUNT = 0
$ LOOP:
$       COUNT = COUNT + 1
$       IF P'COUNT' .EQS. "" THEN EXIT
$       APPEND/NEW &P'COUNT' SAVE.ALL
$       DELETE &P'COUNT';*
$       IF COUNT .LT. 8 THEN GOTO LOOP
$ EXIT
```

This command procedure, COPYDEL.COM, appends files (specified as parameters) to a file called SAVE.ALL. After a file has been appended, the command procedure deletes the file. Up to eight file names can be passed to the command procedure. The file names are assigned to the symbols P1, P2, and so on.

The command procedure uses a counter to refer to parameters that are passed to it. Each time through the loop, the procedure uses an IF command to check whether the value of the current parameter is a null string. When the IF command is scanned, the current value of the symbol COUNT is concatenated with the letter P. The first time through the loop, the IF command tests P1; the second time through the loop it tests P2, and so on. After the expression P‘COUNT’ is evaluated, the substitution of the file names that correspond to P1, P2, and so on is automatic within the context of the IF command.

The APPEND and DELETE commands do not perform any substitution automatically, because they expect and require file specifications as input parameters. The ampersand (&) precedes the P‘COUNT’ expression for these commands to force the appropriate symbol substitution. When these commands are initially scanned each time through the loop, COUNT is substituted with its current value. Then, when the commands execute, the ampersand causes another substitution: the first file specification is substituted for P1, the second file specification is substituted for P2, and so on.

To invoke this procedure, use the following command:

```
$ @COPYDEL ALAMO.TXT BEST.DOC
```

The files ALAMO.TXT and BEST.DOC are each appended to the file SAVE.ALL and are then deleted.

3.
```
$ A = 25
$ CODE = 4 + F$INTEGER("6") - A
$ SHOW SYMBOL CODE
  CODE = -15   HEX = FFFFFFF1   Octal = 1777761
```

This example contains two assignment statements. The first assignment statement assigns the value 25 to the symbol A. The second assignment statement evaluates an expression containing an integer (4), a lexical function (F$INTEGER(‘‘6’’)), and the symbol A. The result of the expression, –15, is assigned to the symbol CODE.

4.
```
$ FILENAME = "JOBSEARCH" - "JOB"
$ FILETYPE = ".OBJ"
$ FILESPEC = FILENAME + FILETYPE
$ TYPE 'FILESPEC'
```

The first command in this example assigns the symbol FILENAME the value ‘‘SEARCH’’. Notice that the string ‘‘SEARCH’’ is the result of the string reduction operation performed by the expression. The second command assigns the symbol FILETYPE the character string ‘‘.OBJ’’. The symbols

FILENAME and FILETYPE are then added together in an expression assigned to the symbol FILESPEC. Because the values of the symbols FILENAME and FILETYPE are concatenated, the resultant value assigned to FILESPEC is the character string ''SEARCH.OBJ''. The symbol FILESPEC is then used as a parameter for the TYPE command. The single quotation marks (') request the command interpreter to replace the symbol FILESPEC with its value SEARCH.OBJ. Thus, the TYPE command types the file named SEARCH.OBJ.

5.
```
$ BELL[0,32] = %X07
$ SHOW SYMBOL BELL
  BELL = ""
```

In this example, the symbol BELL is created with an arithmetic overlay assignment statement. Because the symbol BELL is previously undefined, the hexadecimal value 7 is inserted over a null character string and is interpreted as the ASCII code for the bell character on a terminal. When you issue the command SHOW SYMBOL BELL, the terminal beeps.

If the symbol BELL had been previously defined with an integer value, the result of displaying BELL would have been to show its new integer value.

6.
```
$ $=34
%DCL-W-NOCOMD, no command on line - reenter with alphabetic first
character
$ $$=34
$ SHOW SYMBOL $$
%DCL-W-UNDSYM, undefined symbol - check validity and spelling
$ SHOW SYMBOL $
$ = 34   Hex = 00000022  Octal = 00000000042
```

If you begin a symbol name with the dollar sign ($), use two dollar signs ($$) because DCL discards the first instance of the dollar sign.

:= (String Assignment)

Defines a symbolic name for a character string value.

Format

symbol-name :=[=] string

symbol-name[offset,size] :=[=] replacement-string

Note

Digital advises against assigning a symbolic name that is already a DCL command name. Digital especially discourages the assignment of symbols such as IF, THEN, ELSE, and GOTO, which can affect the interpretation of command procedures.

Parameters

symbol-name

Specifies a string of 1 to 255 characters for the symbol name. The name can contain any alphanumeric characters from the DEC Multinational character set, the underscore (_), and the dollar sign ($). However, the name must begin *only* with an alphabetic character, an underscore, or a dollar sign. Using one equal sign (:=) places the symbol name in the local symbol table for the current command level. Using two equal signs (:==) places the symbol name in the global symbol table.

string

Names the character string value to be equated to the symbol. The string can contain any alphanumeric or special characters. DCL uses a buffer that is 1024 bytes long to hold a string assignment statement. Therefore, the length of the symbol name, the string, and any symbol substitution within the string cannot exceed 1024 characters.

With the string assignment statement (:=), you do not need to enclose a string literal in quotation marks (‘‘ ’’). String values are converted to uppercase automatically. Also, any leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space.

It is easier to use the assignment statement (=) to create symbols with string values because the assignment statement does not automatically convert letters to uppercase and remove extra spaces. Also, the assignment statement allows you to perform string operations in expressions.

To prohibit uppercase conversion and to retain required space and tab characters in a string, place quotation marks around the string. To use quotation marks in a string, enclose the entire string within quotation marks and use a double set of quotation marks within the string. For example:

```
$ TEST := "this     is a ""test"" string"
$ SHOW SYMBOL TEST
  TEST = "this     is a "test" string"
```

In this example, the spaces, lowercase letters, and quotation marks are preserved in the symbol definition.

To continue a symbol assignment on more than one line, use the hyphen (-) as a continuation character. For example:

```
$ LONG_STRING := THIS_SYMBOL_ASSIGNMENT_IS_A_VERY_LONG-
_$ _SYMBOL_STRING
```

To assign a null string to a symbol by using the string assignment statement, do not specify a string. For example:

```
$ NULL :=
```

Specify the string as a string literal, or as a symbol or lexical function that evaluates to a string literal. If you use symbols or lexical functions, place single quotation marks (‘ ’) around them to request symbol substitution. See the *OpenVMS User's Manual* for more information on symbol substitution.

You can also use the string assignment statement to define a foreign command. See the *OpenVMS User's Manual* for more information about foreign commands.

[offset,size]

Specifies that a portion of a symbol value is to be overlaid with a replacement string. This form of the string assignment statement evaluates the value assigned to a symbol and then replaces the portion of the value (defined by the offset and size) with the replacement string. The brackets are required notation, and no spaces are allowed between the symbol name and the left bracket.

The offset specifies the character position relative to the beginning of the symbol name's string value at which replacement is to begin. Offset values start at 0.

If the offset is greater than the offset of the last character in the string you are modifying, spaces are inserted between the end of the string and the offset where the replacement string is added. The maximum offset value you can specify is 768.

The size specifies the number of characters to replace. Size values start at 1.

Specify the offset and size as integer expressions. See the *OpenVMS User's Manual* for more information on integer expressions. The value of the size plus the offset must not exceed 769.

replacement-string

Specifies the string that is used to overwrite the string you are modifying. If the replacement string is shorter than the size argument, the replacement string is filled with blanks on the right until it equals the specified size. Then the replacement string overwrites the string assigned to the symbol name. If the replacement string is longer than the size argument, then the replacement string is truncated on the right to the specified size.

You can specify the replacement string as a string literal, or as a symbol or lexical function that evaluates to a string literal. If you use symbols or lexical functions, place single quotation marks (‘ ’) around them to request symbol substitution. For more information on symbol substitution, see the *OpenVMS User's Manual*.

Examples

1.
```
$ TIME := SHOW TIME
$ TIME
14-DEC-1995 11:55:44
```

   In this example, the symbol TIME is equated to the command string SHOW TIME. Because the symbol name appears as the first word in a command string, the command interpreter automatically substitutes it with its string value and executes the command SHOW TIME.

2.
```
$ STAT := $DBA1:[CRAMER]STAT
$ STAT
```

   This example shows how to define STAT as a foreign command. The symbol STAT is equated to a string that begins with a dollar sign followed by a file specification. The command interpreter assumes that the file specification is that of an executable image, that is, a file with a file type of .EXE. The symbol STAT in this example becomes a synonym for the following command:

```
$ RUN DBA1:[CRAMER]STAT.EXE
```

   When you subsequently enter STAT, the command interpreter executes the image.

3.
```
$ A = "this is a big     space."
$ SHOW SYMBOL A
  A = "this is a big     space."
$ B := 'A'
$ SHOW SYMBOL B
  B = "THIS IS A BIG SPACE."
```

   This example compares the assignment and the string assignment statements. The symbol A is defined using the assignment statement, so lowercase letters and multiple spaces are retained. The symbol B is defined using the string assignment statement. Note that the single quotation marks (') are required; otherwise, the symbol name B would have been equated to the literal string A. However, when symbol A's value is assigned to symbol B, the letters are converted to uppercase and multiple spaces are compressed.

4.
```
$ FILE_NAME := MYFILE
$ FILE_NAME[0,2]:= OL
$ SHOW SYMBOL FILE_NAME
  FILE_NAME = "OLFILE"
```

   In this example, the substring expression in the assignment statement overlays the first 2 characters of the string assigned to the symbol FILE_ NAME with the letters OL. The offset of 0 requests that the overlay begin with the first character in the string, and the size specification of 2 indicates the number of characters to overlay.

5.
```
$ FILE_NAME := MYFILE
$ FILE_TYPE := .TST
$ FILE_NAME[F$LENGTH(FILE_NAME),4] := 'FILE_TYPE'
$ SHOW SYMBOL FILE_NAME
  FILE_NAME = "MYFILE.TST"
```

   In this example, the symbol name FILE_NAME is equated to the string MYFILE and the symbol name FILE_TYPE is equated to the string .TST. The third assignment statement uses the lexical function F$LENGTH to define

the offset value where the overlay is to begin. The symbol name FILE_TYPE is used to refer to the replacement string (.TST). Note that you must use single quotation marks (‘ ’) to request symbol substitution.

The F$LENGTH lexical function returns the length of the string equated to the symbol FILE_NAME; this length is used as the offset. The expression requests that 4 characters of the string currently equated to the symbol FILE_TYPE be placed at the end of the string currently equated to FILE_ NAME. The resultant value of the symbol FILE_NAME is MYFILE.TST.

@ (Execute Procedure)

Executes a command procedure or requests the command interpreter to read subsequent command input from a specific file or device.

Format

@ filespec [parameter[,...]]

Parameters

filespec
Specifies either the input device or the file for the preceding command, or the command procedure to be executed. The default file type is COM. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

parameter[,...]
Specifies from one to eight optional parameters to pass to the command procedure. The symbols (P1, P2, . . . P8) are assigned character string values in the order of entry. The symbols are local to the specified command procedure. Separate each parameter with one or more blanks. Use two consecutive quotation marks ("") to specify a null parameter. You can specify a parameter with a character string value containing alphanumeric or special characters, with the following restrictions:

- The command interpreter converts alphabetic characters to uppercase and uses blanks to delimit each parameter. To pass a parameter that contains embedded blanks or literal lowercase letters, place the parameter in quotation marks.
- If the first parameter begins with a slash (/), you must enclose the parameter in quotation marks (‘‘ ’’).
- To pass a parameter that contains literal quotation marks and spaces, enclose the entire string in quotation marks and use two consecutive quotation marks within the string. For example, the command procedure TEST.COM contains the following line:

  ```
  $ WRITE SYS$OUTPUT P1
  ```

  Enter the following at the DCL prompt ($):

  ```
  $ @TEST "Never say ""quit"""
  ```

  When the procedure TEST.COM executes, the parameter P1 is equated to the following string:

  Never say "quit"

  If a string contains quotation marks and does not contain spaces, the quotation marks are preserved in the string and the letters within the quotation marks remain in lowercase. For example, enter the following at the DCL prompt:

  ```
  $ @TEST abc"def"ghi
  ```

When the procedure TEST.COM executes, the parameter P1 is equated to the following string:

ABC"def"GHI

To use a symbol as a parameter, enclose the symbol in single quotation marks (‘ ’) to force symbol substitution. For example:

```
$ NAME = "JOHNSON"
$ @INFO 'NAME'
```

The single quotation marks cause the value ‘‘JOHNSON’’ to be substituted for the symbol NAME. Therefore, the parameter ‘‘JOHNSON’’ is passed as P1 to INFO.COM.

Description

Use the @ command to execute a command procedure that contains the following:

- DCL command lines or data, or both
- Qualifiers or parameters, or both, for a specific command line

To execute a command procedure containing commands or data, or both, place the @ command at the beginning of a command line and then specify the name of the command procedure file. The command procedure can contain DCL commands and input data for a command or program that is currently executing. All DCL commands in a command procedure must begin with a dollar sign ($). If a command is continued with a hyphen (-), the subsequent lines must not begin with a dollar sign.

Any line in a command procedure that does not contain a dollar sign in the first character position (and is not a continuation line) is treated as input data for the command or program that is currently executing. The DECK command allows you to specify that data contains dollar signs in record position one.

A command procedure can also contain the @ command to execute another command procedure. The maximum command level you can achieve by nesting command procedures is 16, including the top-level command procedure. Command procedures can also be queued for processing as batch jobs, either by using the SUBMIT command or by placing a deck of cards containing the command procedure in the system card reader.

To execute a command procedure that contains qualifiers or parameters, or both, for a specific command line, place the @ command where the qualifiers or parameters normally would be in the command line. Then specify the name of the command procedure file containing the qualifiers or parameters.

If the command procedure file begins with parameters for the command, the @ command must be preceded by a space. For example:

```
$ CREATE TEST.COM
TIME
Ctrl/Z
$ SHOW @TEST
  14-DEC-1994 17:20:26
```

If the file begins with qualifiers for the command, do *not* precede the @ command with a space. For example:

```
$ CREATE TEST_2.COM
/SIZE
```

```
Ctrl/Z
$ DIR@TEST_2

Directory WORK$:[SCHEDULE]

JANUARY.TXT;8         14-DEC-1994 15:47:45.57
FEBRUARY.TXT;7        14-DEC-1994 15:43:16.20
MARCH.TXT;6           14-DEC-1994 11:11:45.74
   .
   .
   .
Total of 11 files.
```

If the file contains parameters or qualifiers, or both, do *not* begin the lines in the file with dollar signs. Any additional data on the command line following @filespec is treated as parameters for the procedure.

Qualifier

/OUTPUT=filespec

Specifies the name of the file to which the command procedure output is written. By default, the output is written to the current SYS$OUTPUT device. The default output file type is .LIS. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the output file specification. System responses and error messages are written to SYS$COMMAND as well as to the specified file. The /OUTPUT qualifier must immediately follow the file specification of the command procedure; otherwise, the qualifier is interpreted as a parameter to pass to the command procedure.

You can also redefine SYS$OUTPUT to redirect the output from a command procedure. If you place the following command as the first line in a command procedure, output will be directed to the file you specify:

```
$ DEFINE SYS$OUTPUT filespec
```

When the procedure exits, SYS$OUTPUT will be restored to its original equivalence string. This produces the same result as using the /OUTPUT qualifier when you execute the command procedure.

Examples

1.
```
$ CREATE DOFOR.COM
$ ON WARNING THEN EXIT
$ IF P1.EQS."" THEN INQUIRE P1 FILE
$ FORTRAN/LIST 'P1'
$ LINK 'P1'
$ RUN 'P1'
$ PRINT 'P1'
Ctrl/Z
$ @DOFOR AVERAGE
```

This example shows a command procedure, named DOFOR.COM, that executes the FORTRAN, LINK, and RUN commands to compile, link, and execute a program. The ON command requests that the procedure not continue if any of the commands result in warnings or errors.

When you execute DOFOR.COM, you can pass the file specification of the FORTRAN program as the parameter P1. If you do not specify a value for P1 when you execute the procedure, the INQUIRE command issues a prompting message to the terminal and equates what you enter with the symbol P1. In this example, the file name AVERAGE is assigned to P1. The file type is not

included because the commands FORTRAN, LINK, RUN, and PRINT provide default file types.

2.
```
$ @MASTER/OUTPUT=MASTER.LOG
```

This command executes a procedure named MASTER.COM; all output is written to the file MASTER.LOG.

3.
```
$ CREATE FILES.COM
*.FOR, *.OBJ
Ctrl/Z
$ DIRECTORY @FILES
```

This example shows a command procedure, FILES.COM, that contains parameters for a DCL command line. You can execute this procedure after the DIRECTORY command to get a listing of all Fortran source and object files in your current default directory.

4.
```
$ CREATE QUALIFIERS.COM
/DEBUG/SYMBOL_TABLE/MAP/FULL/CROSS_REFERENCE
Ctrl/Z
$ LINK SYNAPSE@QUALIFIERS
```

This example shows a command procedure, QUALIFIERS.COM, that contains qualifiers for the LINK command. When you enter the LINK command, specify the command procedure immediately after the file specification of the file you are linking. Do not type a space between the file specification and the @ command.

5.
```
$ CREATE SUBPROCES.COM
$ RUN 'P1' -
  /BUFFER_LIMIT=1024 -
  /FILE_LIMIT=4 -
  /PAGE_FILES=256 -
  /QUEUE_LIMIT=2 -
  /SUBPROCESS_LIMIT=2 -
  'P2'  'P3'  'P4'  'P5'  'P6'  'P7'  'P8'
Ctrl/Z
$ @SUBPROCES  LIBRA  /PROCESS_NAME=LIBRA
```

This example shows a command procedure named SUBPROCES.COM. This procedure issues the RUN command to create a subprocess to execute an image and also contains qualifiers defining quotas for subprocess creation. The name of the image to be run is passed as the parameter P1. Parameters P2 to P8 can be used to specify additional qualifiers.

In this example, the file name LIBRA is equated to P1; it is the name of an image to execute in the subprocess. The qualifier /PROCESS_NAME=LIBRA is equated to P2; it is an additional qualifier for the RUN command.

6.
```
$ CREATE EDOC.COM
$ ASSIGN SYS$COMMAND:  SYS$INPUT
$ NEXT:
$      INQUIRE NAME "File name"
$      IF NAME.EQS."" THEN EXIT
$      EDIT/TPU 'NAME'.DOC
$      GOTO NEXT
```

```
Ctrl/Z
$ @EDOC
```

This procedure, named EDOC.COM, invokes the EVE editor. When an edit session is terminated, the procedure loops to the label NEXT. Each time through the loop, the procedure requests another file name for the editor and supplies the default file type .DOC. When a null line is entered in response to the INQUIRE command, the procedure terminates with the EXIT command.

The ASSIGN command changes the equivalence name of SYS$INPUT for the duration of the procedure. This change allows the EVE editor to read input data from the terminal, rather than from the command procedure file (the default input data stream if SYS$INPUT had not been changed). When the command procedure exits, SYS$INPUT is reassigned to its original value.

ACCOUNTING

Runs the Accounting utility, which produces reports of resource use. For a complete description of the Accounting utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

ACCOUNTING [filespec[,...]]

ALLOCATE

Provides your process with exclusive access to a device until you deallocate the device or terminate your process. Optionally associates a logical name with the device.

Requires read (R), write (W), or control access.

Format

ALLOCATE device-name[:][,...] [logical-name[:]]

Parameters

device-name[:][,...]
Specifies the name of a physical device or a logical name that translates to the name of a physical device. The device name can be generic: if no controller or unit number is specified, any device that satisfies the specified part of the name is allocated. If more than one device is specified, the first available device is allocated.

logical-name[:]
Specifies a string of 1 to 255 alphanumeric characters. Enclose the string in single quotation marks (') if it contains blanks. Trailing colons (:) are not used. The name becomes a process logical name with the device name as the equivalence name. The logical name remains defined until it is explicitly deleted or your process terminates.

Qualifiers

/GENERIC
/NOGENERIC (default)
Indicates that the first parameter is a device *type* rather than a device *name*. Example device types are: RX50, RD52, TK50, RC25, RCF25, and RL02. The first free, nonallocated device of the specified name and type is allocated.

The /[NO]GENERIC qualifier is placed before the *device-name* parameter in the ALLOCATE command line. For example, you can allocate an RK07 device by entering the following command at the DCL prompt ($):

```
$ ALLOCATE/GENERIC RK07 DISK
```

The following table shows some device types that you can specify with the /GENERIC qualifier. To see what devices are available, refer to your SPD for the OpenVMS version they are currently using,

Devices by Classification				
Disk Devices				
EF51	EF58	ESE20	ESE58	EZ31
EZ58	HSZ10	HSZ40	ML11	RA60
RA70	RA80	RA90	RAH72	RB02
RB80	RC25	RCF25	RD26	RD54

Devices by Classification				
Disk Devices				
RF30	RFH73	RK07	RL02	RM80
RP07	RX18	RX50	RZ01	RZF01
Compact Disk Devices				
RRD40	RRD40S	RRD42	RRD43	RRD44
RRD50	RV20	RV60	RV80	RW504
RW510	RW514	RW516	RWZ01	RWZ21
RWZ31	RWZ51	RWZ52	RWZ53	RWZ54
Tape Devices				
TA78	TA90	TA90E	TA91	TAD85
TAPE9	TD34	TD44	TE16	TF30
TF70	TK50	TK60	TK70	TKZ60
TL810	TLZ7	TM32	TS11	TSZ08
TU45	TU81	TZ30	TZ895	TZX0

/LOG (default)
/NOLOG
Displays a message indicating the name of the device allocated. If the operation specifies a logical name that is currently assigned to another device, then the superseded value is displayed.

Examples

1.
```
$ ALLOCATE  DMB2:
%DCL-I-ALLOC, _DMB2: allocated
```

The ALLOCATE command in this example requests the allocation of a specific RK06/RK07 disk drive, that is, unit 2 on controller B. The system response indicates that the device was allocated successfully.

2.
```
$ ALLOCATE  MT,MF:   TAPE:
%DCL-I-ALLOC, _MTB2: allocated
.
.
.
$ SHOW LOGICAL TAPE:
TAPE: = _MTB2:    (process)
$ DEALLOCATE TAPE:
$ DEASSIGN TAPE:
```

The ALLOCATE command in this example requests the allocation of a tape device whose name begins with MT or MF and assigns it the logical name TAPE. The ALLOCATE command locates an available tape device whose name begins with MT, and responds with the name of the device allocated. (If no tape device beginning with MT had been found, the ALLOCATE command would have searched for a device beginning with MF.) Subsequent references to the device TAPE in user programs or command strings are translated to the device name MTB2.

When the tape device is no longer needed, the DEALLOCATE command deallocates it and the DEASSIGN command deletes the logical name. Note that the logical name TAPE was specified with a colon on the ALLOCATE command, but that the logical name table entry does not have a colon.

3.
```
$ ALLOCATE/GENERIC RL02 WORK
%DCL-I-ALLOC, _DLA1: allocated
%DCL-I-SUPERSEDE, previous value of WORK has been superseded
```

The ALLOCATE command in this example requests the allocation of any RL02 disk device and assigns the logical name WORK to the device. The completion message identifies the allocated device and indicates that the assignment of the logical name WORK supersedes a previous assignment of that name.

4.
```
$ ALLOCATE $TAPE1
%DCL-I-ALLOC, _MUA0: allocated
```

The ALLOCATE command in this example allocates the tape device MUA0, which is associated with the logical name $TAPE1.

5.
```
$ ALLOCATE /GENERIC RX50 ACCOUNTS
```

The ALLOCATE command in this example allocates the first free floppy disk drive and makes its name equivalent to the process logical name ACCOUNTS.

ANALYZE/AUDIT

Invokes the Audit Analysis utility, which selectively extracts and displays information from security audit log files or security archive files. For a complete description of the Audit Analysis utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

ANALYZE/AUDIT [filespec]

ANALYZE/CRASH_DUMP

Invokes the System Dump Analyzer utility, which analyzes a system dump file. The /CRASH_DUMP qualifier is required.

For a complete description of the System Dump Analyzer utility on Alpha, see the *OpenVMS Alpha System Dump Analyzer Utility Manual*. For a complete description of the System Dump Analyzer utility on VAX, see the *OpenVMS VAX System Dump Analyzer Utility Manual*.

Format

ANALYZE/CRASH_DUMP filespec

ANALYZE/DISK_STRUCTURE

Invokes the Analyze/Disk_Structure utility, which does the following:

- Checks the readability and validity of Files–11 On-Disk Structure Level 1 and Files–11 On-Disk Structure Level 2 disk volumes.
- Reports errors and inconsistencies.

The /DISK_STRUCTURE qualifier is required. For a complete description of the Analyze/Disk_Structure utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

ANALYZE/DISK_STRUCTURE device-name[:]

ANALYZE/ERROR_LOG

Invokes the Errorlog Report Formatter, which selectively reports the contents of an error log file. The /ERROR_LOG qualifier is required. For a complete description of the Error Log utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

ANALYZE/ERROR_LOG [filespec[,...]]

ANALYZE/IMAGE

Analyzes the contents of an executable image file or a shareable image file on Alpha, VAX, and translated VAX images, and checks for obvious errors in the image file. The /IMAGE qualifier is required. For general information about image files, see the description of the linker in the *OpenVMS Linker Utility Manual*. (Use the ANALYZE/OBJECT command to analyze the contents of an object file.)

Note

The OpenVMS VAX Version 6.1 and 6.2 ANALYZE/IMAGE command cannot analyze an OpenVMS Alpha image.

Format

ANALYZE/IMAGE filespec[,...]

Parameter

filespec[,...]
Specifies the name of one or more image files that you want analyzed. You must specify at least one file name. If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+). The default file type is .EXE.

The asterisk (*) and percent sign (%) wildcard characters are allowed in the file specification.

Description

The ANALYZE/IMAGE command provides a description of the components of an executable image file or shareable image file. It also verifies that the structure of the major parts of the image file is correct. However, the ANALYZE/IMAGE command cannot ensure that program execution is error free.

If errors are found, the first error of the worst severity is returned. For example, if a warning (A) and two errors (B and C) are found, the first error (B) is returned as the image exit status. The image exit status is placed in the DCL symbol $STATUS at image exit.

The ANALYZE/IMAGE command distinguishes Alpha system image files from VAX system image files by examining the extended image header (EIHD).

The ANALYZE/IMAGE command provides the following information:

- Image type—Identifies whether the image is executable or shareable.
- Image transfer addresses—Identify the addresses to which control is passed at image execution time.
- Image version—Identifies the revision level (major ID and minor ID) of the image.
- Location and size of the image's symbol vector (Alpha only)

- Location of the debugger symbol table (DST)—Identifies the location of the DST in the image file. DST information is present only in executable images that have been linked with the /DEBUG or the /TRACEBACK command qualifier.
- Location of the global symbol table (GST)—Identifies the location of the GST in the image file. GST information is present only in shareable image files.
- Patch information—Indicates whether the image has been patched (changed without having been recompiled or reassembled and relinked). If a patch is present, the actual patch code can be displayed.
- Image section descriptors (ISD)—Identify portions of the image binary contents that are grouped in VMScluster systems according to their attributes. An ISD contains information that the image activator needs when it initializes the address space for an image. For example, an ISD tells whether the ISD is shareable, whether it is readable or writable, whether it is based or position independent, and how much memory should be allocated.
- Fixup vectors—Contain information that the image activator needs to ensure the position independence of shareable image references.
- System version categories—For an image that is linked against the executive (the system shareable image on Alpha or the system symbol table on VAX), displays both the values of the system version categories for which the image was linked originally and the values for the system that is currently running. You can use these values to identify changes in the system since the image was linked last.

The ANALYZE/IMAGE command has command qualifiers and positional qualifiers. By default, if you do not specify any positional qualifiers (for example, /GST or /HEADER), the entire image is analyzed. If you do specify a positional qualifier, the analysis excludes all other positional qualifiers except the /HEADER qualifier (which is always enabled) and any qualifier that you request explicitly.

Qualifiers

/FIXUP_SECTION

Positional qualifier.

Specifies that the analysis should include all information in the fixup section of the image.

If you specify the /FIXUP_SECTION qualifier after the ANALYZE/IMAGE command, the fixup section of each image file in the parameter list is analyzed.

If you specify the /FIXUP_SECTION qualifier after a file specification, only the information in the fixup section of that image file is analyzed.

/GST

Positional qualifier.

Specifies that the analysis should include all global symbol table records. This qualifier is valid only for shareable images.

If you specify the /GST qualifier after the ANALYZE/IMAGE command, the global symbol table records of each image file in the parameter list are analyzed.

If you specify the /GST qualifier after a file specification, only the global symbol table records of that file are analyzed.

/HEADER
Positional qualifier.

Specifies that the analysis should include all header items and image section descriptions. The image header items are always analyzed.

/INTERACTIVE
/NOINTERACTIVE (default)
Specifies whether the analysis is interactive. In interactive mode, as each item is analyzed, the results are displayed on the screen and you are asked whether you want to continue.

/OUTPUT=filespec
Identifies the output file for storing the results of the image analysis. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification. If you specify a file type and omit the file name, the default file name ANALYZE is used. The default file type is .ANL. If you omit the qualifier, the results are output to the current SYS$OUTPUT device.

/PATCH_TEXT
Positional qualifier.

Specifies that the analysis include all patch text records. If you specify the /PATCH_TEXT qualifier after the ANALYZE/IMAGE command, the patch text records of each image file in the parameter list are analyzed.

If you specify the /PATCH_TEXT qualifier after a file specification, only the patch text records of that file are analyzed.

Examples

1.
```
$ ANALYZE/IMAGE  LINEDT
```

   The ANALYZE/IMAGE command in this example produces a description and an error analysis of the image LINEDT.EXE. Output is sent to the current SYS$OUTPUT device. By default, the entire image is analyzed.

2.
```
$ ANALYZE/IMAGE/OUTPUT=LIALPHEX/FIXUP_SECTION/PATCH_TEXT  LINEDT, ALPRIN
```

   The ANALYZE/IMAGE command in this example produces a description and an error analysis of the fixup sections and patch text records of LINEDT.EXE and ALPRIN.EXE in file LIALPHEX.ANL. Output is sent to the file LIALPHEX.ANL.

ANALYZE/MEDIA

Invokes the Bad Block Locator utility, which analyzes block-addressable devices and records the location of blocks that cannot reliably store data. For a complete description of the Bad Block Locator utility, see the *OpenVMS Bad Block Locator Utility Manual*.

Format

ANALYZE/MEDIA device

ANALYZE/OBJECT

Analyzes the contents of an object file. Also, checks for any obvious errors. The /OBJECT qualifier is required. (Use the ANALYZE/IMAGE command to analyze the contents of an image file.)

Format

ANALYZE/OBJECT filespec[,...]

Parameter

filespec[,...]
Specifies the object files or object module libraries you want analyzed (the default file type is .OBJ). Use commas (,) or plus signs (+) to separate file specifications. The asterisk (*) and the percent sign (%) wildcard characters are allowed in the file specification.

Description

The ANALYZE/OBJECT command describes the contents of one or more object modules contained in one or more files. It also performs a partial error analysis. This analysis determines whether all records in an object module conform in content, format, and sequence to the specifications of the Alpha or VAX Object Language.

The ANALYZE/OBJECT command automatically distinguishes Alpha objects from VAX objects by examining the format of the extended object modules header (EOMH).

ANALYZE/OBJECT is intended primarily for programmers of compilers, debuggers, or other software involving the operating system's object modules. It checks that the object language records generated by the object modules are acceptable to the Linker utility, and it identifies certain errors in the file. It also provides a description of the records in the object file or object module library. For more information on the linker and on the Alpha and VAX object languages, refer to the *OpenVMS Linker Utility Manual*.

The ANALYZE/OBJECT command analyzes the object modules in order, record by record, from the first to the last record in the object module. Fields in each record are analyzed in order from the first to the last field in the record. After the object module is analyzed, you should compare the content and format of each type of record to the required content and format of that record as described by the OpenVMS Alpha or OpenVMS VAX Object Language. This comparison is particularly important if the analysis output contains a diagnostic message.

Linking an object module differs from analyzing an object module. Object language commands are not executed in an analysis, but they are executed in a linking operation. As a result, even if the analysis is error free, the linking operation may not be. In particular, the analysis does not check the following:

- That data arguments in TIR commands are in the correct format.
- That ''Store Data'' TIR commands are storing within legal address limits.

Therefore, as a final check, you should still link an object module whose analysis is error free.

If an error is found, however, the first error of the worst severity that is discovered is returned. For example, if a warning (A) and two errors (B and C) are signaled, then the first error (B) is returned as the image exit status, which is placed in the DCL symbol $STATUS at image exit.

ANALYZE/OBJECT uses positional qualifiers; that is, qualifiers whose function depends on their position in the command line. When a positional qualifier precedes all of the input files in a command line, it affects all input files. For example, the following command line requests that the analysis include the global symbol directory records in files A, B, and C:

```
$ ANALYZE/OBJECT/GSD A,B,C
```

Conversely, when a positional qualifier is associated with only one file in the parameter list, only that file is affected. For example, the following command line requests that the analysis include the global symbol directory records in file B only:

```
$ ANALYZE/OBJECT A,B/GSD,C
```

Typically, all records in an object module are analyzed. However, when the /DBG, /EOM, /GSD, /LNK, /MHD, /TBT, or /TIR qualifier is specified, only the record types indicated by the qualifiers are analyzed. All other record types are ignored.

By default, the analysis includes all record types unless you explicitly request a limited analysis using appropriate qualifiers.

Note

End-of-module (EOM) records and module header (MHD) records are always analyzed, no matter which qualifiers you specify.

Qualifiers

/DBG

Positional qualifier.

Specifies that the analysis should include all debugger information records. If you want the analysis to include debugger information for all files in the parameter list, insert the /DBG qualifier immediately following the /OBJECT qualifier. If you want the analysis to include debugger information selectively, insert the /DBG qualifier immediately following each of the selected file specifications.

/EOM

Positional qualifier.

Specifies that the analysis should be limited to MHD records, EOM records, and records explicitly specified by the command. If you want this to apply to all files in the parameter list, insert the /EOM qualifier immediately following the /OBJECT qualifier.

To make the /EOM qualifier applicable selectively, insert it immediately following each of the selected file specifications.

Note

End-of-module records can be EOM or EOMW records. See the *OpenVMS Linker Utility Manual* for more information.

/GSD
Positional qualifier.

Specifies that the analysis should include all global symbol directory (GSD) records.

If you want the analysis to include GSD records for each file in the parameter list, specify the /GSD qualifier immediately following the /OBJECT qualifier.

If you want the analysis to include GSD records selectively, insert the /GSD qualifier immediately following each of the selected file specifications.

/INCLUDE[=(module[,...])]
When the specified file is an object module library, use this qualifier to list selected object modules within the library for analysis. If you omit the list or specify an asterisk (*), all modules are analyzed. If you specify only one module, you can omit the parentheses.

/INTERACTIVE
/NOINTERACTIVE (default)
Controls whether the analysis occurs interactively. In interactive mode, as each record is analyzed, the results are displayed on the screen, and you are asked whether you want to continue.

/LNK
Positional qualifier.

Specifies that the analysis should include all link option specification (LNK) records.

If you want the analysis to include LNK records for each file in the parameter list, specify the /LNK qualifier immediately following the /OBJECT qualifier.

If you want the analysis to include LNK records selectively, insert the /LNK qualifier immediately following each of the selected file specifications.

/MHD
Positional qualifier.

Specifies that the analysis should be limited to MHD records, EOM records, and records explicitly specified by the command. If you want this analysis to apply to all files in the parameter list, insert the /MHD qualifier immediately following the /OBJECT qualifier.

To make the /MHD qualifier applicable selectively, insert immediately following each of the selected file specifications.

/OUTPUT[=filespec]
Directs the output of the object analysis (the default is SYS$OUTPUT). If you specify a file type and omit the file name, the default file name ANALYZE is used. The default file type is ANL.

The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

/TBT
Positional qualifier.

Specifies that the analysis should include all module traceback (TBT) records.

If you want the analysis to include TBT records for each file in the parameter list, specify the /TBT qualifier immediately following the /OBJECT qualifier.

If you want the analysis to include TBT records selectively, insert the /TBT qualifier immediately following each of the selected file specifications.

/TIR
Positional qualifier.

Specifies that the analysis should include all text information and relocation (TIR) records.

If you want the analysis to include TIR records for each file in the parameter list, specify the /TIR qualifier immediately following the /OBJECT qualifier.

If you want the analysis to include TIR records selectively, insert the /TIR qualifier immediately following the selected file specifications.

Examples

1. `$ ANALYZE/OBJECT/INTERACTIVE  LINEDT`

   In this example, the ANALYZE/OBJECT command produces a description and a partial error analysis of the object file LINEDT.OBJ. By default, all types of records are analyzed. Output is to the terminal, because the /INTERACTIVE qualifier has been used. As each item is analyzed, the utility displays the results on the screen and asks if you want to continue.

2. `$ ANALYZE/OBJECT/OUTPUT=LIOBJ/DBG  LINEDT`

   In this example, the ANALYZE/OBJECT command analyzes only the debugger information records of the file LINEDT.OBJ. Output is to the file LIOBJ.ANL.

ANALYZE/PROCESS_DUMP

Invokes the OpenVMS Debugger to analyze a process dump file that was created when an image failed during execution. (Use the /DUMP qualifier with the RUN or the SET PROCESS command to generate a dump file.)

The ANALYZE/PROCESS_DUMP command can display a process dump file for either an Alpha or VAX image. For a complete description of the debugger (including information about the DEBUG command), see the *OpenVMS Debugger Manual*.

Requires read (R) access to the dump file.

Format

ANALYZE/PROCESS_DUMP dump-file

Parameter

dump-file
Specifies the dump file to be analyzed with the debugger.

Description

The ANALYZE/PROCESS_DUMP command examines the dump file of an image that failed during execution. The OpenVMS Debugger is invoked automatically. To cause a dump file to be created for a process, you must use the /DUMP qualifier with the RUN command when invoking the image, or you must use the SET PROCESS/DUMP command before invoking the image.

Analysis of a process dump on a system other than on the system where the dump was generated is not recommended. Different configurations can cause the process executing the ANALYZE/PROCESS_DUMP command to fail to successfully load the dumped image. For example, if the systems have different versions of the operating system, the analysis may work, but it is not guaranteed.

Other restrictions include the configuration of the control regions (P1 space), the process running at the time of the dump, and the process performing the ANALYZE/PROCESS_DUMP command. The location of the base of the user stack for each process, which depends upon the size of P1 space, determines whether the processes are compatible. The size of P1 space for the process analyzing the dump must be less than the size of P1 space for the process which created the dump. If you are analyzing the dump on a different system, but with the same version of the operating system, you can decrease the size of P1 space by modifying one or more of the system parameters that affect the size of P1 space. You can modify the system parameter IMGIOCNT dynamically.

Alpha

On Alpha, in some cases, the OpenVMS Debugger is incapable of analyzing the dumped image. For example, when the dumped image's PC is set to an invalid address or when the dumped image's stack is corrupted by a bad process descriptor, you must use the Delta Debugger (DELTA) to analyze the dump. To use DELTA as the debugger, the image SYS$LIBRARY:DELTA must be installed by invoking the Install utility. For complete information on the Install utility, see the *OpenVMS System Management Utilities Reference Manual*.♦

Qualifiers

/FULL
Displays all known information about the failing process.

/IMAGE=image-name
/NOIMAGE
Specifies the image whose symbols are to be used in analyzing the dump. If you use the /NOIMAGE qualifier, no symbols are taken from any image. By default, symbols are taken from the image with the same name as the image that was running at the time of the dump.

/INTERACTIVE
/NOINTERACTIVE (default)
Causes the display of information to pause when your terminal screen is filled. Press the Return key to display additional information. By default, the display is continuous.

/MISCELLANEOUS
Displays process information and registers at the time of the dump. Refer to the $GETJPI system service for further explanation of the process information displayed.

/OUTPUT=filespec
Writes the information to the specified file. By default, the information is written to the current SYS$OUTPUT device. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

/RELOCATION
Displays the addresses to which data structures saved in the dump are mapped in P0 space. (Examples of such data structures are the stacks.) The data structures in the dump must be mapped into P0 space so that the debugger can use those data structures in P1 space.

Examples

VAX

1.
```
$ ANALYZE/PROCESS/FULL ZIPLIST

 R0 = 00018292  R1 = 8013DE20  R2 = 7FFE6A40   R3 = 7FFE6A98
 R4 = 8013DE20  R5 = 00000000  R6 = 7FFE7B9A   R7 = 0000F000
 R8 = 00000000  R9 = 00000000  R10 = 00000000  R11 = 00000000
 SP = 7FFAEF44  AP = 7FFAEF48  FP  = 7FFAEF84
 FREE_P0_VA  00001600    FREE_P1_VA  7FFAC600
 Active ASTs  00         Enabled ASTs 0F
 Current Privileges  FFFFFF80  1010C100
 Event Flags  00000000  E0000000
 Buffered I/O count/limit 6/6
 Direct I/O count/limit   6/6
 File count/limit         27/30
 Process count/limit      0/0
 Timer queue count/limit  10/10
 AST count/limit          6/6
 Enqueue count/limit      30/30
 Buffered I/O total 7      Direct I/O total 18
```

```
 Link Date  27-DEC-1994 15:02:00.48   Patch Date  17-NOV-1994 00:01:53.71
 ECO Level  0030008C  00540040  00000000  34303230
 Kernel stack 00000000 pages at 00000000 moved to 00000000
 Exec stack 00000000 pages at 00000000 moved to 00000000
 Vector page 00000001 page at 7FFEFE00 moved to 00001600
 PIO (RMS) area 00000005 pages at 7FFE1200 moved to 00001800
 Image activator context 00000001 page at 7FFE3400 moved to 00002200
 User writable context 0000000A pages at 7FFE1C00 moved to 00002400
Creating a subprocess
         VAX DEBUG Version 5.4
DBG>
```

This example shows the output of the ANALYZE/PROCESS command when used with the /FULL qualifier on a VAX system. The file specified, ZIPLIST, contains the dump of a process that encountered a fatal error. The DBG> prompt indicates that the debugger is ready to accept commands.♦

Alpha

2.
```
$ ANALYZE/PROCESS/FULL ARITH.DMP

 R0  = 000000000000000F R1  = 0000000000000001 R2  = 0000000000010000
 R3  = 000000007FF41E14 R4  = 000000007FFBF80C R5  = 000000007FFBF91C
 R6  = 000000007FFA0D34 R7  = 000000007FFA0D34 R8  = 000000007FFA05F8
 R9  = 000000007FFA0800 R10 = 000000007FFA1380 R11 = 000000007FFBE3E0
 R12 = 0000000000000004 R13 = FFFFFFFF84EF3730 R14 = 0000000000000000
 R15 = 0000000000000000 R16 = 00000000002D4EC8 R17 = 0000000100000000
 R18 = 00000000002D4EAE R19 = 0000000000000000 R20 = 0000000000000000
 R21 = 0000000000000000 R22 = 002D559100000000 R23 = 00000000002D5591
 R24 = 002D559100000000 R25 = 0000000000000003 R26 = 0000000000020040
 R27 = 0000000000032AA0 R28 = 0000000000052DA0 AP  = 000000007FE6F6F0
 FP  = 000000007FE6FA20 SP  = 000000007FE6FA20 PC  = 0000000000020048

 FREE_P0_VA  002F6000  FREE_P1_VA  7FE68000
 Active ASTs  00  Enabled ASTs 0F
 Current Privileges  00000000  00108000
 Event Flags  00000000  E0000000
 Buffered I/O count/limit 100/100
 Direct I/O count/limit   100/100
 File count/limit         98/100
 Process count/limit      0/0
 Timer queue count/limit  10/10
 AST count/limit          98/100
 Enqueue count/limit      600/600
 Buffered I/O total 118     Direct I/O total 37

 Link Date   14-DEC-1994 13:06:06.72

 Kernel stack 00000000 pagelets at 00000000 moved to 00000000
 Exec stack 00000000 pagelets at 00000000 moved to 00000000
 Vector page 00000002 pagelet at 7FFF0000 moved to 002F6000
 PIO (RMS) area 00000000 pagelets at 00000000 moved to 00000000
 Image activator context 00000001 pagelet at 7FFD0E00 moved to 002F8000
 User writeable context 00000008 pagelets at 7FFC0000 moved to 002FA000

Condition signalled to take dump:
%SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000,
-Fmask=00000001, summary=04, PC=00020048, PS=0000001B
-SYSTEM-F-FLTDIV, arithmetic trap, floating/decimal divide by zero
-at PC=00020048, PS=0000001B

         OpenVMS Alpha DEBUG Version T1.0-FT4
%DEBUG-I-NOLOCALS, image does not contain local symbols
%DEBUG-I-NOGLOBALS, some or all global symbols not accessible
%DEBUG-I-NOUNIVERSALS, shareable image contains no universal symbols
%DEBUG-I-INITIAL, language is UNKNOWN, module set to SHARE$ARITH
DBG>
```

This example shows the output of the ANALYZE/PROCESS command when used with the /FULL qualifier on an Alpha system.♦

Alpha

3.
```
$ INSTALL
INSTALL> ADD SYS$LIBRARY:DELTA
INSTALL> EXIT
$ DEFINE LIB$DEBUG DELTA
$ ANALYZE/PROCESS/FULL PCFS_SERVER.DMP

 R0  = 004558452E524556 R1  = 00000000004F1080  R2  = 000000000000048C
 R3  = 0000000000000000 R4  = 0000000000000000  R5  = 0000000000000001
 R6  = 0000000000F380C0 R7  = 00000000000000E9  R8  = 00000000002D3F70
 R9  = 0000000000342FB8 R10 = 00000000002CE330  R11 = 0000000000000047
 R12 = 00000000005107C0 R13 = 0000000000F38140  R14 = 0000000000D5D5C8
 R15 = 00000000000000F0 R16 = 000000000003793C  R17 = 00000000004F108C
 R18 = 0000000000000000 R19 = 00000000004F1C00  R20 = 00000000004F1C00
 R21 = FFFFFFFF8052C4B6 R22 = 0000000000000001  R23 = FFFFFFFF80042E90
 R24 = 0000000000000001 R25 = 0000000000000000  R26 = 0000000000002000
 R27 = 00000000000184A0 R28 = 0000000000000001  AP  = 0000000000F37E10
 FP  = 0000000000DE0ED8 SP  = 0000000000F38110  PC  = 0000000000002000

 FREE_P0_VA  00F50000           FREE_P1_VA  7FE6A000
 Active ASTs  00                Enabled ASTs 0F
 Current Privileges  FFFFFFFF  FFF7FFFF
 Event Flags  80000000  E0000000
 Buffered I/O count/limit 32758/32767
 Direct I/O count/limit   32767/32767
 File count/limit         32720/32767
 Process count/limit      0/0
 Timer queue count/limit  32763/32767
 AST count/limit          32754/32767
 Enqueue count/limit      32760/32767
 Buffered I/O total 17353137            Direct I/O total 333631

 Link Date   8-SEP-1992 16:42:37.94

 Kernel stack 00000020 pagelets at 7FF92000 moved to 00F50000
 Exec stack 00000020 pagelets at 7FF96000 moved to 00F54000
 Vector page 00000002 pagelet at 7FFF0000 moved to 00F58000
 PIO (RMS) area 00000007 pagelets at 7FFD0000 moved to 00F5A000
 Image activator context 00000001 pagelet at 7FFD0E00 moved to 00F5C000
 User writeable context 00000008 pagelets at 7FFC0000 moved to 00F5E000

Condition signalled to take dump:
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=00002000, PC=00002000, PS=0000001B

AlphaVMS DELTA Version 1.5

Exception - Signal reason = 0000000C

00002000!
```

In this example, the SYS$LIBRARY:DELTA image is installed on an Alpha system by invoking the Install utility. Next, the logical LIB$DEBUG is defined as DELTA in order to use the Delta Debugger (DELTA). The ANALYZE/PROCESS_DUMP command displays exception information and then a prompt. You can analyze the stack or other information provided to you in the dump.♦

ANALYZE/RMS_FILE

Invokes the Analyze/RMS_File utility, which is used to inspect and analyze the internal structure of an OpenVMS RMS file. The /RMS_FILE qualifier is required. For a complete description of the Analyze/RMS_File utility, see the *OpenVMS Record Management Utilities Reference Manual*.

Format

ANALYZE/RMS_FILE filespec[,...]

ANALYZE/SYSTEM

Invokes the System Dump Analyzer utility, which analyzes a running system. The /SYSTEM qualifier is required. For a complete description of the System Dump Analyzer utility on Alpha, see the *OpenVMS Alpha System Dump Analyzer Utility Manual*. For a complete description of the System Dump Analyzer utility on VAX, see the *OpenVMS VAX System Dump Analyzer Utility Manual*.

Format

ANALYZE/SYSTEM

APPEND

Adds the contents of one or more specified input files to the end of the specified output file.

Format

APPEND input-filespec[,...] output-filespec

Parameters

input-filespec[,...]
Specifies the names of one or more input files to be appended. Multiple input files are appended to the output file in the order specified. If you specify more than one input file, separate each file specification with either a comma (,) or a plus sign (+).

The asterisk (*) and the percent sign (%) wildcard characters are allowed in the input file specifications.

output-filespec
Specifies the name of the file to which the input files will be appended.

You must specify at least one field in the output file specification. If you do not specify a device or directory, the APPEND command uses the current default device and directory. Other unspecified fields default to the corresponding fields of the first input file specification.

If you use the asterisk (*) wildcard character in any fields of the output file specification, the APPEND command uses the corresponding field of the input file specification. If you are appending more than one input file, the APPEND command uses the corresponding fields from the first input file.

Description

The APPEND command is similar in syntax and function to the COPY command. Normally, the APPEND command adds the contents of one or more files to the end of an existing file without incrementing the version number. The /NEW_VERSION qualifier causes the APPEND command to create a new output file if no file with that name exists.

Note that there are special considerations for using the APPEND command with DECwindows compound documents. For more information, see the *Guide to OpenVMS File Applications*.

Qualifiers

/ALLOCATION=number-of-blocks
Forces the initial allocation of the output file to the specified number of 512-byte blocks. If you do not specify the /ALLOCATION qualifier, or if you specify it without the *number-of-blocks* parameter, the initial allocation of the output file is determined by the size of the input file.

The allocation size is applied only if a new file is actually created by using the /NEW_VERSION qualifier.

/BACKUP

Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /BACKUP qualifier selects files according to the dates of their most recent backups. This qualifier is incompatible with the /CREATED, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /BEFORE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

Specify the UIC by using standard UIC format as described in the *OpenVMS User's Manual*.

/CONFIRM
/NOCONFIRM (default)

Controls whether a request is issued before each append operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	Ctrl/Z
1	0	ALL
	Return	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers include: NO, FALSE, 0, and pressing the Return key. Entering QUIT or pressing Ctrl/Z indicates that you want to stop processing the command at that point. When you respond by entering ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CONTIGUOUS
/NOCONTIGUOUS

Specifies that the output file must occupy physically contiguous disk blocks. By default, the APPEND command creates an output file in the same format as the corresponding input file and does not report an error if not enough space exists for a contiguous allocation. This qualifier is relevant only with the /NEW_VERSION qualifier.

If an input file is contiguous, the APPEND command attempts to create a contiguous output file, but does not report an error if there is not enough space. If you append multiple input files of different formats, the output file may or may not be contiguous. Use the /CONTIGUOUS qualifier to ensure that the output file is contiguous.

/CREATED (default)
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the /BACKUP, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/EXCLUDE=(filespec[,...])
Excludes the specified files from the append operation. You can include a directory but not a device in the file specification. Wildcard characters (* and %) are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /EXPIRED qualifier selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the /BACKUP, /CREATED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/EXTENSION=number-of-blocks
Specifies the number of blocks to be added to the output file each time the file is extended. When you specify the /EXTENSION qualifier, the /NEW_VERSION qualifier is assumed and need not be typed on the command line. This qualifier is relevant only with the /NEW_VERSION qualifier.

The extension value is applied only if a new file is actually created.

/LOG
/NOLOG (default)
Controls whether the APPEND command displays the file specifications of each file appended. If the /LOG qualifier is specified, the command displays the file specifications of the input and output files as well as the number of blocks or records appended after each append operation.

/MODIFIED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /MODIFIED qualifier selects files according to the dates on which they were last modified. This qualifier is incompatible with the /BACKUP, /CREATED, and /EXPIRED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time modifiers, the default is the /CREATED qualifier.

/NEW_VERSION
/NONEW_VERSION (default)
Controls whether the APPEND command creates a new output file if the specified output file does not exist. (By default, the specified output file already exists.) If

the specified output file does not already exist, use the /NEW_VERSION qualifier to create a new output file. If the output file does exist, the /NEW_VERSION qualifier is ignored and the input file is appended to the output file.

/PROTECTION=(ownership[:access][,...])
Specifies protection for the output file.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), write (W), execute (E), or delete (D).

The default protection, including any protection attributes not specified, is that of the existing output file. If no output file exists, the current default protection applies. This qualifier is relevant only with the /NEW_VERSION qualifier.

For more information on specifying protection codes, see the *OpenVMS Guide to System Security*.

/READ_CHECK
/NOREAD_CHECK (default)
Reads each record in the input files twice to verify that it has been read correctly.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /SINCE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/WRITE_CHECK
/NOWRITE_CHECK (default)
Reads each record in the output file after the record is written to verify that it was appended successfully and that the output file can subsequently be read without error.

Examples

1.
```
$ APPEND  TEST3.DAT TESTALL.DAT
```

The APPEND command appends the contents of the file TEST3.DAT from the default disk and directory to the file TESTALL.DAT, also located on the default disk and directory.

2.
```
$ APPEND/NEW_VERSION/LOG *.TXT   MEM.SUM
%APPEND-I-CREATED, USE$:[MAL]MEM.SUM;1 created
%APPEND-S-COPIED, USE$:[MAL]A.TXT;2 copied to USE$:[MAL]MEM.SUM;1 (1 block)
%APPEND-S-APPENDED, USE$:[MAL]B.TXT;3 appended to USE$:[MAL]MEM.SUM;1 (3 records)
%APPEND-S-APPENDED, USE$:[MAL]G.TXT;7 appended to USE$:[MAL]MEM.SUM;1 (51 records)
```

The APPEND command appends all files with file types of .TXT to a file named MEM.SUM. The /LOG qualifier requests a display of the specifications of each input file appended. If the file MEM.SUM does not exist, the APPEND command

creates it, as the output shows. The number of blocks or records shown in the output refers to the source file and not to the target file total.

3.
```
$ APPEND/LOG A.DAT, B.MEM   C.*
%APPEND-S-APPENDED, USE$:[MAL]A.DAT;4 appended to USE$:[MAL]C.DAT;4 (2 records)
%APPEND-S-APPENDED, USE$:[MAL]B.MEM;5 appended to USE$:[MAL]C.DAT;4 (29 records)
```

The APPEND command appends the files A.DAT and B.MEM to the file C.DAT, which must already exist.

4.
```
$ APPEND/LOG A.*   B.*
%APPEND-S-APPENDED, USE$:[MAL]A.DAT;5 appended to USE$:[MAL]B.DAT;1 (5 records)
%APPEND-S-APPENDED, USE$:[MAL]A.DOC;2 appended to USE$:[MAL]B.DAT;1 (1 record)
```

Both the input and output file specifications contain wildcard characters in the file type field. The APPEND command appends each file with a file name of A to an existing file with B as its file name. The file type of the first input file located determines the output file type.

5.
```
$ APPEND BOSTON"JOHN_SMITH YANKEE"::DEMO01.DAT, DEMO2.DAT
$ _To:   DALLAS::DISK1:[MODEL.TEST]TEST.DAT
```

This APPEND command adds the contents of the files DEMO1.DAT and DEMO2.DAT at remote node BOSTON to the end of the file TEST.DAT at remote node DALLAS.

ASSIGN

Creates a logical name and assigns an equivalence string, or a list of strings, to the specified logical name. If you specify an existing logical name, the new equivalence name replaces the existing equivalence name.

Format

ASSIGN equivalence-name[,...] logical-name[:]

Parameters

equivalence-name[,...]

Specifies a character string of 1 to 255 characters. Defines the equivalence name, usually a file specification, device name, or other logical name, to be associated with the logical name in the specified logical name table. If the string contains other than uppercase alphanumeric, dollar sign ($), or underscore (_) characters, enclose it in quotation marks (‘‘ ’’). Use two sets of quotation marks (‘‘‘‘ ’’’’) to denote an actual quotation mark within the string. Specifying more than one equivalence name for a logical name creates a search list.

When you specify an equivalence name that will be used as a file specification, you must include the punctuation marks (colons (:), brackets ([ ]), and periods (.)) that would be required if the equivalence name were used directly as a file specification. Therefore, if you specify a device name as an equivalence name, terminate the device name with a colon.

The ASSIGN command allows you to assign the same logical name to more than one equivalence name. When you specify more than one equivalence name for a logical name, you create a search list. For more information on search lists, see the *OpenVMS User’s Manual*.

logical-name[:]

Specifies the logical name string, which is a character string containing up to 255 characters. You choose a logical name to represent the equivalence name in the specified logical name table.

If the string contains other than uppercase alphanumeric, dollar sign, or underscore characters, enclose it in quotation marks. Use two sets of quotation marks to denote an actual quotation mark. If you terminate the logical-name parameter with a colon, the system removes the colon before placing the name in a logical name table. (This differs from the DEFINE command, which saves the colon.) If the logical name is to be entered into the process directory (LNM$PROCESS_DIRECTORY) or system directory (LNM$SYSTEM_DIRECTORY) logical name tables, then the name may only have from 1 to 31 alphanumeric characters (including the dollar sign and underscore). By default, the logical name is placed in the process logical name table.

If the logical name contains any characters other than alphanumeric characters, the dollar sign, or the underscore, enclose the name in quotation marks. If the logical name contains quotation marks, enclose the name in quotation marks and use two sets of quotation marks in the places where you want one set of quotation marks to occur. Note that if you enclose a name in quotation marks, the case of alphabetic characters is preserved.

ASSIGN

Description

The ASSIGN command creates an entry in a logical name table by defining a logical name to stand for one or more equivalence names. An equivalence name can be a device name, another logical name, a file specification, or any other string.

To specify the logical name table where you want to enter a logical name, use the /PROCESS, /JOB, /GROUP, /SYSTEM, or /TABLE qualifier. If you enter more than one of these qualifiers, only the last one entered is accepted. If you do not specify a table, the default is /TABLE=LNM$PROCESS (or /PROCESS).

To specify the access mode of the logical name you are creating, use the /USER_MODE, the /SUPERVISOR_MODE, or the /EXECUTIVE_MODE qualifier. If you enter more than one of these qualifiers, only the last one entered is accepted. If you do not specify an access mode, then a supervisor-mode name is created. You can create a logical name in the same mode as the table in which you are placing the name or in an outer mode. (User mode is the outermost mode; executive mode is the innermost mode.)

You can enter more than one logical name with the same name in the same logical name table, as long as each name has a different access mode. (However, if an existing logical name within a table has the NO_ALIAS attribute, you cannot use the same name to create a logical name in an outer mode in this table.)

If you create a logical name with the same name, in the same table, and in the same mode as an existing name, the new logical name assignment replaces the existing assignment.

You can also use the DEFINE command to create logical names. To delete a logical name from a table, use the DEASSIGN command.

Note

Avoid assigning a logical name that matches the file name of an executable image in SYS$SYSTEM:. Such an assignment will prohibit you from invoking that image.

For additional information on how to create and use logical names, see the *OpenVMS User's Manual*.

Qualifiers

/EXECUTIVE_MODE

Requires SYSNAM (system logical name) privilege.

Creates an executive-mode logical name. If you specify executive mode, but do not have SYSNAM privilege, a supervisor-mode logical name is created. The mode of the logical name must be the same as or external to (less privileged than) the mode of the table in which you are placing the name.

/GROUP

Requires SYSPRV (system privilege) or GRPNAM (group logical name) privilege.

Places the logical name in the group logical name table. Other users who have the same group number in their user identification codes (UICs) can access the logical name. The /GROUP qualifier is synonymous with the /TABLE=LNM$GROUP qualifier.

/JOB
Places the logical name in the jobwide logical name table. All processes within the same job tree as the process creating the logical name can access the logical name. The /JOB qualifier is synonymous with the /TABLE=LNM$JOB qualifier.

/LOG (default)
/NOLOG
Displays a message when a new logical name supersedes an existing name.

/NAME_ATTRIBUTES[=(keyword[,...])]
Specifies the attributes for a logical name. By default, no attributes are set. You can specify the following keywords for attributes:

CONFINE	Does not copy the logical name into a spawned subprocess; this keyword is relevant only for logical names in a private table.
NO_ALIAS	Prohibits creation of logical names with the same name in an outer (less privileged) access mode within the specified table. If another logical name with the same name and an outer access mode already exists in this table, the name is deleted.

If you specify only one keyword, you can omit the parentheses. Only the attributes you specify are set.

/PROCESS (default)
Places the logical name in the process logical name table. The /PROCESS qualifier is synonymous with the /TABLE=LNM$PROCESS qualifier.

/SUPERVISOR_MODE (default)
Creates a supervisor-mode logical name in the specified table.

/SYSTEM
Requires SYSNAM (system logical name) or SYSPRV (system privilege) privilege.

Places the logical name in the system logical name table. All system users can access the logical name. The /SYSTEM qualifier is synonymous with the /TABLE=LNM$SYSTEM qualifier.

/TABLE=name
Requires write (W) access to the table if the table is shareable.

Specifies the logical name table in which the logical name is to be entered. You can use the /TABLE qualifier to specify a user-defined logical name table (created with the CREATE/NAME_TABLE command); to specify the process, job, group, or system logical name tables; or to specify the process or system logical name directory tables.

If you specify the table name using a logical name that has more than one translation, the logical name is placed in the first table found. For example, if you specify ASSIGN/TABLE=LNM$FILE_DEV and LNM$FILE_DEV is equated to LNM$PROCESS, LNM$JOB, LNM$GROUP, and LNM$SYSTEM, then the logical name is placed in LNM$PROCESS.

If you do not explicitly specify the /TABLE qualifier, the default is the /TABLE=LNM$PROCESS qualifier.

/TRANSLATION_ATTRIBUTES[=(keyword[,...])]
Equivalence-name qualifier.

Specifies attributes of the equivalence-name parameter. Possible keywords are as follows:

CONCEALED — Indicates that the equivalence string is the name of a concealed device.

When a concealed device name is defined, the system displays the logical name, rather than the equivalence string, in messages that refer to the device. If you specified the CONCEALED attribute, then the equivalence string must be a physical device name.

TERMINAL — Indicates that the equivalence string should not be translated iteratively; logical name translation should terminate with the current equivalence string.

If you specify only one keyword, you can omit the parentheses. Only the attributes you specify are set.

Note that different equivalence strings of the same logical name can have different translation attributes specified.

/USER_MODE
Creates a user-mode logical name in the specified table.

If you specify a user-mode logical name in the process logical name table, that logical name is used for the execution of a single image only; user-mode entries are deleted from the logical name table when any image executing in the process exits; that is, after any DCL command that executes an image or user program completes execution. Also, user-mode logical names are automatically deleted when invoking and exiting a command procedure.

Examples

1.
```
$ ASSIGN $DISK1:[ACCOUNTS.MEMOS] MEMOSD
```

   The ASSIGN command in this example equates the partial file specification $DISK1:[ACCOUNTS.MEMOS] to the logical name MEMOSD.

2.
```
$ ASSIGN/USER_MODE $DISK1:[ACCOUNTS.MEMOS]WATER.TXT TM1
```

   The ASSIGN command in this example equates the logical name TM1 to a file specification. After the next image runs, the logical name is deassigned automatically.

3.
```
$ ASSIGN XXX1:[CHARLES]  CHARLIE
$ PRINT  CHARLIE:TEST.DAT
Job 274 entered on queue SYS$PRINT
```

   The ASSIGN command in this example associates the logical name CHARLIE with the directory name [CHARLES] on the disk XXX1. Subsequent references to the logical name CHARLIE result in the correspondence between the logical name CHARLIE and the disk and directory specified. The PRINT command queues a copy of the file XXX1:[CHARLES]TEST.DAT to the system printer.

4.
```
$ ASSIGN YYY2:  TEMP:
$ SHOW LOGICAL TEMP
   "TEMP" = "YYY2:" (LNM$PROCESS_TABLE)
$ DEASSIGN TEMP
```

The ASSIGN command in this example equates the logical name TEMP to the device YYY2. TEMP is created in supervisor mode and placed in the process logical name table. The SHOW LOGICAL command verifies that the logical name assignment was made. Note that the logical name TEMP was terminated with a colon in the ASSIGN command, but that the command interpreter deleted the colon before placing the name in the logical name table. Thus, you can specify TEMP without a colon in the subsequent DEASSIGN command. You should omit the colon in the SHOW LOGICAL command (for example, SHOW LOGICAL TEMP).

5.
```
$ MOUNT TTT1: MASTER TAPE
$ ASSIGN TAPE:NAMES.DAT PAYROLL
$ RUN PAYROLL
   .
   .
   .
```

The MOUNT command in this example establishes the logical name TAPE for the device TTT1, which has the volume labeled MASTER mounted on it. The ASSIGN command equates the logical name PAYROLL with the file named NAMES.DAT on the logical device TAPE. Thus, an OPEN request in a program referring to the logical name PAYROLL results in the correspondence between the logical name PAYROLL and the file NAMES.DAT on the tape whose volume label is MASTER.

6.
```
$ CREATE/NAME_TABLE TABLE1
$ ASSIGN/TABLE=LNM$PROCESS_DIRECTORY TABLE1,-
_$ LNM$PROCESS,LNM$JOB,LNM$GROUP,LNM$SYSTEM LNM$FILE_DEV
$ ASSIGN/TABLE=TABLE1 -
_$ /TRANSLATION_ATTRIBUTES=CONCEALED  DBA1:  WORK_DISK
```

The CREATE/NAME_TABLE command in this example creates the process private logical name table TABLE1.

The first ASSIGN command ensures that TABLE1 is searched first in any logical name translation of a file specification or device name (because TABLE1 is the first item in the equivalence string for the logical name LNM$FILE_DEV, which determines the default search sequence of logical name tables whenever a device or file specification is translated).

The second ASSIGN command assigns the logical name WORK_DISK to the physical device DBA1, and places the name in TABLE1. The logical name has the concealed attribute. Therefore, the logical name WORK_DISK will be displayed in system messages.

7.
```
$ ASSIGN/TABLE=LNM$PROCESS/TABLE=LNM$GROUP  DBA0:  SYSFILES
$ SHOW LOGICAL  SYSFILES
  "SYSFILES" = "DBA0:" (LNM$GROUP_000240)
```

The ASSIGN command in this example contains conflicting qualifiers. When you specify conflicting qualifiers, the ASSIGN command uses the last qualifier specified. The response from the SHOW LOGICAL command indicates that the name was placed in the group logical name table.

8.
```
$ ASSIGN/TABLE=LNM$GROUP 'F$TRNLNM("SYS$COMMAND")' TERMINAL
%DCL-I-SUPERSEDE, previous value of TERMINAL has been superseded
```

   The ASSIGN command in this example uses the lexical function F$TRNLNM to translate the logical name SYS$COMMAND and use the result as the equivalence name for the logical name TERMINAL. The message from the ASSIGN command indicates that an entry for the logical name TERMINAL already existed in the group logical name table, and that the new entry has replaced the previous one.

   If this command is used in a LOGIN.COM file, the entry for TERMINAL will be redefined at the beginning of each terminal session. The current process and any subprocesses it creates can execute images that use the logical name TERMINAL to write messages to the current terminal device.

9.
```
$ ASSIGN DALLAS::DMA1:  DATA
```

   The ASSIGN command in this example associates the logical name DATA with the device specification DMA1 on remote node DALLAS. Subsequent references to the logical name DATA result in references to the disk on the remote node.

10.
```
$ CREATE AVERAGE.COM
$ ASSIGN/USER_MODE SYS$COMMAND:  SYS$INPUT
$ EDIT/EDT AVERAGE.FOR
$ FORTRAN AVERAGE
$ LINK AVERAGE
$ RUN AVERAGE
87
80
90
9999
$ EXIT
Ctrl/Z
$ @AVERAGE.COM
```

    The CREATE command in this example creates the command procedure AVERAGE.COM. Then the command procedure is executed.

    The command procedure uses the ASSIGN command with the /USER_MODE qualifier to change temporarily the value of SYS$INPUT. When the EDT editor is invoked, it accepts input from the terminal. This allows you to create or modify the program AVERAGE.FOR interactively.

    When you exit from EDT, SYS$INPUT is reassigned to its original value (the input stream provided by the command procedure). Thus, when the program AVERAGE.FOR is ready to accept input, it looks for that input in the command procedure.

ASSIGN/MERGE

Removes all jobs from one queue and merges them into another existing queue. This command does not affect jobs that are executing.

Requires manage (M) access to both queues.

Format

ASSIGN/MERGE target-queue[:] source-queue[:]

Parameters

target-queue[:]
Specifies the name of the queue into which the jobs are being merged.

source-queue[:]
Specifies the name of the queue from which the jobs are being removed.

Description

The ASSIGN/MERGE command removes the pending jobs in one queue and places them in another queue. This command does not affect any executing jobs in either the target queue or the source queue. Jobs currently running in the source queue complete in that queue. This command is generally used with printer queues, although it can be used with batch queues.

The ASSIGN/MERGE command is particularly useful when a line printer malfunctions. By entering the ASSIGN/MERGE command, you can reroute existing jobs to a different printing device. To perform the merge operation without losing or disrupting any jobs, stop the source queue with the STOP /QUEUE/NEXT command. Then enter the STOP/QUEUE/REQUEUE command to ensure that the current job on the source queue is requeued for processing on the target queue. (If the STOP/QUEUE/REQUEUE command fails to requeue the job, use the STOP/QUEUE/RESET command to regain control of the queue.) Once you enter the STOP commands, enter the ASSIGN/MERGE command.

Example

```
$ STOP/QUEUE/NEXT LPB0
$ STOP/QUEUE/REQUEUE=LPA0 LPB0
$ ASSIGN/MERGE LPA0 LPB0
```

In this example, the STOP/QUEUE/NEXT command prevents another job from executing on queue LPB0. The STOP/QUEUE/REQUEUE command requeues the current job running on LPB0 to the target queue LPA0. The ASSIGN/MERGE command removes the remaining jobs from the LPB0 printer queue and places them in the LPA0 printer queue.

ASSIGN/QUEUE

Assigns, or redirects, a logical queue to a single execution queue. The ASSIGN /QUEUE command can be used only with printer or terminal queues.

Requires manage (M) access to both queues.

Format

ASSIGN/QUEUE queue-name[:] logical-queue-name[:]

Parameters

queue-name[:]
Specifies the name of the execution queue. The queue cannot be a logical queue, a generic queue, or a batch queue.

logical-queue-name[:]
Specifies the name of the logical queue.

Description

The ASSIGN/QUEUE command sets up a one-to-one correspondence between a logical queue and an execution queue. Jobs submitted to the logical queue are always queued to the specified execution queue for eventual printing.

When you enter the ASSIGN/QUEUE command, the logical queue cannot be running.

Once you initialize a logical queue, use the ASSIGN/QUEUE command to associate the logical queue with an existing execution queue. You must perform the following tasks to set up a logical queue:

1. Initialize the logical queue with the INITIALIZE/QUEUE command. (Do not use the /START qualifier.)
2. Assign the logical queue name to an existing execution queue.
3. Start the logical queue with the START/QUEUE command.

After you enter the START/QUEUE command for the logical queue, jobs can be sent to the logical queue for processing.

Examples

1.
```
$ INITIALIZE/QUEUE/DEFAULT=FLAG=ONE/START LPA0
$ INITIALIZE/QUEUE TEST_QUEUE
$ ASSIGN/QUEUE LPA0 TEST_QUEUE
$ START/QUEUE TEST_QUEUE
```

This example first initializes and starts the printer queue LPA0. The LPA0 queue is set to have a flag page precede each job. The second INITIALIZE/QUEUE command creates the logical queue TEST_QUEUE. The ASSIGN/QUEUE command assigns the logical queue TEST_QUEUE to the printer queue LPA0. The START/QUEUE command starts the logical queue.

2. `$ INITIALIZE/QUEUE/START LPB0`

   The ASSIGN/QUEUE command is not needed in this example because a logical queue is not being initialized. A printer queue is being initialized; LPB0 is the name of a line printer. After you enter the INITIALIZE/QUEUE /START command, jobs can be queued to LPB0 for printing.

ATTACH

Transfers control from your current process (which then hibernates) to the specified process.

The ATTACH and SPAWN commands cannot be used if your terminal has an associated mailbox.

Format

ATTACH [process-name]

Parameter

process-name

Specifies the name of a parent process or spawned subprocess to which control passes. The process must already exist, be part of your current job, and share the same input stream as your current process. However, the process cannot be your current process or a subprocess created with the /NOWAIT qualifier.

Process names can contain from 1 to 15 alphanumeric characters. If a connection to the specified process cannot be made, an error message is displayed.

The *process-name* parameter is incompatible with the /IDENTIFICATION qualifier.

Description

The ATTACH command allows you to connect your input stream to another process. You can use the ATTACH command to change control from one subprocess to another subprocess or to the parent process.

When you enter the ATTACH command, the parent or ''source'' process is put into hibernation, and your input stream is connected to the specified destination process. You can use the ATTACH command to connect to a subprocess that is part of a current job left hibernating as a result of the SPAWN/WAIT command or another ATTACH command as long as the connection is valid. (No connection can be made to the current process, to a process that is not part of the current job, or to a process that does not exist. If any of these connections are attempted, an error message is displayed.)

You can also use the ATTACH command in conjunction with the SPAWN /WAIT command to return to a parent process without terminating the created subprocess. See the description of the SPAWN command for more details.

Qualifier

/IDENTIFICATION=pid

Specifies the process identification (PID) of the process to which terminal control will be transferred. Leading zeros can be omitted. The /IDENTIFICATION qualifier is incompatible with the *process-name* parameter.

If you omit the /IDENTIFICATION qualifier, you must specify a process name.

Examples

1. `$ ATTACH JONES_2`

   The ATTACH command transfers the terminal's control to the subprocess JONES_2.

2. `$ ATTACH/IDENTIFICATION=30019`

   The ATTACH command switches control from the current process to a process having the PID 30019. Notice that because the /IDENTIFICATION qualifier is specified, the *process-name* parameter is omitted.

BACKUP

Invokes the Backup utility (BACKUP) to perform one of the following backup operations:

- Make copies of disk files.
- Save disk files as data in a file created by BACKUP on disk or magnetic tape. (Files created by BACKUP are called save sets.)
- Restore disk files from a BACKUP save set.
- Compare disk files or files in a BACKUP save set with other disk files.
- List information about files in a BACKUP save set to an output device or file.

You cannot invoke BACKUP to back up a system disk; a system disk must be bootstrapped to run. For a complete description of BACKUP and information on backing up the system disk, see the *OpenVMS System Manager's Manual* and the *OpenVMS System Management Utilities Reference Manual*.

Format

BACKUP input-specifier output-specifier

CALL

Transfers control to a labeled subroutine within a command procedure.

Format

CALL label [parameter [...]]

Parameters

label

Specifies a label of 1 to 255 alphanumeric characters that appears as the first item on a command line. A label cannot contain embedded blanks. When the CALL command is executed, control passes to the command following the specified label.

The label can precede or follow the CALL statement in the current command procedure. A label in a command procedure must be terminated with a colon (:). Labels for subroutines must be unique.

Labels declared in inner procedure levels are inaccessible from outer levels, as in the following example:

```
$CALL B
$SUBROUTINE A
$  B: SUBROUTINE
$  ENDSUBROUTINE
$ENDSUBROUTINE
```

In this example, the label B in subroutine A is inaccessible from the outer procedure level.

parameter [...]

Specifies from one to eight optional parameters to pass to the command procedure. Use quotation marks ("") to specify a null parameter. The parameters assign character string values to the symbols named P1, P2, and so on in the order of entry, to a maximum of eight. The symbols are local to the specified command procedure. Separate each parameter with one or more spaces.

You can specify a parameter with a character string value containing alphanumeric or special characters, with the following restrictions:

- The command interpreter converts alphabetic characters to uppercase and uses blanks to delimit each parameter. To pass a parameter that contains embedded blanks or lowercase letters, enclose the parameter in quotation marks (‘‘ ’’).
- If the first parameter begins with a slash (/), you must enclose the parameter in quotation marks.
- To pass a parameter that contains quotation marks and spaces, enclose the entire string in quotation marks and use two sets of quotation marks within the string. For example:

  ```
  $ CALL SUB1 "Never say ""quit"""
  ```

  When control transfers to SUB1, the parameter P1 is equated to the following string:

  ```
  Never say "quit"
  ```

If a string contains quotation marks and does not contain spaces, the quotation marks are preserved in the string and the letters within the quotation marks remain in lowercase. For example:

```
$ CALL SUB2 abc"def"ghi
```

When control transfers to SUB2, the parameter P1 is equated to the string:

ABCdefGHI

To use a symbol as a parameter, enclose the symbol in single quotation marks (‘ ’) to force symbol substitution. For example:

```
$ NAME = "JOHNSON"
$ CALL INFO 'NAME'
```

The single quotation marks cause the value ‘‘JOHNSON’’ to be substituted for the symbol ‘NAME’. Therefore, the parameter ‘‘JOHNSON’’ is passed as P1 to the subroutine INFO.

Description

The CALL command transfers control to a labeled subroutine within a command procedure. The CALL command is similar to the @ (execute procedure) command in that it creates a new procedure level. The advantage of the CALL command is that it does not require files to be opened and closed to process the procedure. Using the CALL command also makes managing a set of procedures easier because they can all exist in one file rather than in several files.

When you use the CALL command to transfer control to a subroutine, a new procedure level is created and the symbols P1 to P8 are assigned the values of the supplied arguments. Execution then proceeds until an EXIT command is encountered. At this point, control is transferred to the command line following the CALL command.

Procedures can be nested to a maximum of 32 levels, which includes any combination of command procedure and subroutine calls. Local symbols and labels defined within a nested subroutine structure are treated the same way as if the routines had been invoked with the @ command; that is, labels are valid only for the subroutine level in which they are defined.

Local symbols defined in an outer subroutine level are available to any subroutine levels at an inner nesting level; that is, the local symbols can be read, but they cannot be written to. If you assign a value to a symbol that is local to an outer subroutine level, a new symbol is created at the current subroutine level. However, the symbol in the outer procedure level is not modified.

The SUBROUTINE and ENDSUBROUTINE commands define the beginning and end of a subroutine. The label defining the entry point to the subroutine must appear either immediately before the SUBROUTINE command or on the same command line.

A subroutine can have only one entry point. The subroutine must begin with the SUBROUTINE command as the first executable statement. If an EXIT command is not specified in the procedure, the ENDSUBROUTINE command functions as an EXIT command.

The SUBROUTINE command performs two different functions depending on the context in which it is executed. If executed as the result of a CALL command, it initiates a new procedure level, defines the parameters P1 to P8 as specified in the CALL statement, and begins execution of the subroutine. If the SUBROUTINE verb is encountered in the execution flow of the procedure without having been invoked by a CALL command, all the commands following the SUBROUTINE command are skipped until the corresponding ENDSUBROUTINE command is encountered.

Note

The SUBROUTINE and ENDSUBROUTINE commands cannot be abbreviated to fewer than 4 characters.

Qualifier

/OUTPUT=filespec

Writes all output to the file or device specified. By default, the output is written to the current SYS$OUTPUT device and the output file type is LIS. System responses and error messages are written to SYS$COMMAND as well as to the specified file. If you specify /OUTPUT, the qualifier must immediately follow the CALL command. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the output file specification.

You can also redefine SYS$OUTPUT to redirect the output from a command procedure. If you place the following command as the first line in a command procedure, output will be directed to the file you specify:

```
$ DEFINE SYS$OUTPUT filespec
```

When the procedure exits, SYS$OUTPUT is restored to its original equivalence string. This produces the same result as using the /OUTPUT qualifier when you execute the command procedure.

Example

```
$
$! CALL.COM
$
$! Define subroutine SUB1
$!
$ SUB1: SUBROUTINE
   .
   .
   .
$ CALL SUB2 !Invoke SUB2 from within SUB1
   .
   .
   .
$ @FILE  !Invoke another procedure command file
   .
   .
   .
$ EXIT
$ ENDSUBROUTINE !End of SUB1 definition
$!
$! Define subroutine SUB2
$!
$ SUB2: SUBROUTINE
   .
   .
   .
$ EXIT
$ ENDSUBROUTINE !End of SUB2 definition
$!
$! Start of main routine. At this point, both SUB1 and SUB2
$! have been defined but none of the previous commands have
$! been executed.
$!
$ START:
$ CALL/OUTPUT=NAMES.LOG SUB1 "THIS IS P1"
   .
   .
   .
$ CALL SUB2 "THIS IS P1" "THIS IS P2"
   .
   .
   .
$ EXIT  !Exit this command procedure file
```

The command procedure in this example shows how to use the CALL command to transfer control to labeled subroutines. The example also shows that you can call a subroutine or another command file from within a subroutine. The CALL command invokes the subroutine SUB1, directing output to the file NAMES.LOG and allowing other users write (W) access to the file. The subroutine SUB2 is called from within SUB1. The procedure executes SUB2 and then uses the @ (execute procedure) command to invoke the command procedure FILE.COM. When all the commands in SUB1 have executed, the CALL command in the main procedure calls SUB2 a second time. The procedure continues until SUB2 has executed.

CANCEL

Cancels wakeup requests for a specified process, including wakeup requests scheduled with either the RUN command or the $SCHDWK system service.

Requires one of the following:

- **Ownership of the process.**
- **GROUP privilege to cancel scheduled wakeup requests for processes in the same group but not owned by you.**
- **WORLD privilege to cancel scheduled wakeup requests for any process in the system.**

Format

CANCEL [[node-name::]process-name]

Parameters

node-name::

The name of the node on which the specified process is running. The node name can have as many as six alphanumeric characters. The two colons (::) count for two additional characters, for a total of eight.

You cannot specify a node name on a different VMScluster system from the current process.

process-name

The name of the process for which wakeup requests are to be canceled. The process name can have up to 15 alphanumeric characters.

The specified process must be in the same group as the current process.

Description

The CANCEL command cancels scheduled wakeup requests for the specified process.

The CANCEL command does not delete the specified process. If the process is executing an image when the CANCEL command is issued for it, the process hibernates instead of exiting after the image completes execution.

To delete a hibernating process for which wakeup requests have been canceled, use the STOP command. You can determine whether a subprocess has been deleted by entering the SHOW PROCESS command with the /SUBPROCESSES qualifier.

A local process name can look like a remote process name. Therefore, if you specify ATHENS::SMITH, the system checks for a process named ATHENS::SMITH on the local node before checking node ATHENS for a process named SMITH.

You also can use the /IDENTIFICATION=pid qualifier to specify a process name. If you use the /IDENTIFICATION qualifier and the *process-name* parameter

together, the qualifier overrides the parameter. If you do not specify either the *process-name* parameter or the /IDENTIFICATION qualifier, the CANCEL command cancels scheduled wakeup requests for the current (that is, the issuing) process.

Qualifier

/IDENTIFICATION=pid

Identifies the process by its process identification (PID). You can omit leading zeros when you specify the PID.

Examples

1.
```
$ CANCEL CALENDAR
```

   The CANCEL command in this example cancels a wakeup request for a process named CALENDAR (which continues to hibernate until it is deleted with the STOP command).

2.
```
$ RUN/SCHEDULE=14:00 STATUS
%RUN-S-PROC_ID, identification of created process is 0013012A
    .
    .
    .
$ CANCEL/IDENTIFICATION=13012A
```

   The RUN command in this example creates a process to execute the image STATUS. The process hibernates and is scheduled to be awakened at 14:00. Before the process is awakened, the CANCEL command cancels the wakeup request.

3.
```
$ RUN/PROCESS_NAME=LIBRA/INTERVAL=1:00    LIBRA
%RUN-S-PROC_ID, identification of created process is 00130027
    .
    .
    .
$ CANCEL LIBRA
$ STOP LIBRA
```

   The RUN command in this example creates a subprocess named LIBRA to execute the image LIBRA.EXE at hourly intervals.

   Subsequently, the CANCEL command cancels the wakeup request. The process continues to exist, but in a state of hibernation, until the STOP command deletes it.

CLOSE

Closes a file opened with the OPEN command and deassigns the associated logical name.

Format

CLOSE logical-name[:]

Parameter

logical-name[:]
Specifies the logical name assigned to the file when it was opened with the OPEN command.

Description

Files that are opened for reading or writing at the command level remain open until closed with the CLOSE command, or until the process terminates. If a command procedure that opens a file terminates without closing the open file, the file remains open; the command interpreter does not automatically close it.

Qualifiers

/ERROR=label
Specifies a label in the command procedure to receive control if the close operation results in an error. Overrides any ON condition action specified. If an error occurs and the target label is successfully given control, the global symbol $STATUS retains the code for the error that caused the error path to be taken.

/LOG (default)
/NOLOG
Generates a warning message when you attempt to close a file that was not opened by DCL. If you specify the /ERROR qualifier, the /LOG qualifier has no effect. If the file has not been opened by DCL, the error branch is taken and no message is displayed.

Examples

1.
```
$ OPEN/READ INPUT_FILE   TEST.DAT
$ READ_LOOP:
$ READ/END_OF_FILE=NO_MORE  INPUT_FILE  DATA_LINE
   .
   .
   .
$ GOTO READ_LOOP
$ NO_MORE:
$ CLOSE INPUT_FILE
```

The OPEN command in this example opens the file TEST.DAT and assigns it the logical name of INPUT_FILE. The /END_OF_FILE qualifier on the READ command requests that, when the end-of-file (EOF) is reached, the command interpreter should transfer control to the line at the label NO_MORE. The CLOSE command closes the input file.

2.
```
$ @READFILE
Ctrl/Y
$ STOP
$ SHOW LOGICAL/PROCESS
    .
    .
    .
  "INFILE" = "_DB1"
  "OUTFILE" = "_DB1"
$ CLOSE INFILE
$ CLOSE OUTFILE
```

In this example, pressing Ctrl/Y interrupts the execution of the command procedure READFILE.COM. Then, the STOP command stops the procedure. The SHOW LOGICAL/PROCESS command displays the names that currently exist in the process logical name table. Among the names listed are the logical names INFILE and OUTFILE, assigned by OPEN commands in the procedure READFILE.COM.

The CLOSE commands close these files and deassign the logical names.

CONNECT

Connects your physical terminal to a virtual terminal that is connected to another process.

You must connect to a virtual terminal that is connected to a process with your user identification code (UIC). No other physical terminals may be connected to the virtual terminal.

Format

CONNECT virtual-terminal-name

Parameter

virtual-terminal-name
Specifies the name of the virtual terminal to which you are connecting. A virtual terminal name always begins with the letters VTA. To determine the name of the virtual terminal that is connected to a process, enter the SHOW USERS command.

Description

The CONNECT command connects you to a separate process, as opposed to the SPAWN and ATTACH commands, which create and attach subprocesses.

The CONNECT command is useful when you are logged in to the system using telecommunications lines. If there is noise over the line and you lose the carrier signal, your process does not terminate. After you log in again, you can reconnect to the original process and log out of your second process.

To use the CONNECT command, the virtual terminal feature must be enabled for your system with the System Manager utility (SYSMAN) on OpenVMS Alpha systems and the System Generation utility (SYSGEN) on OpenVMS VAX systems.

If virtual terminals are allowed on your system, use the SET TERMINAL /DISCONNECT/PERMANENT command to enable the virtual terminal characteristic for a particular physical terminal. When you enable this characteristic, a virtual terminal is created when a user logs in to the physical terminal. The physical terminal is connected to the virtual terminal, which is in turn connected to the process.

For new virtual terminals, you must first set the TT2$V_DISCONNECT bit in the TTY_DEFCHAR2 system parameter and reboot the system. This is done by creating the virtual device VTA0: using the ttdriver. For example:

Alpha

```
$ RUN SYSMAN IO CONNECT/NOADAPTER -
_$ /DRIVER=SYS$LOADABLE_IMAGES:SYS$TTDRIVER VTA0:♦
```

CONNECT

VAX

```
$ RUN SYSGEN CONNECT /NOADAPTER/DRIVER=TTDRIVER VTA0:♦
```

When the connection between the physical terminal and the virtual terminal is broken, you are logged out of your current process (and any images that the process is executing stop running) unless you have specified the /NOLOGOUT qualifier.

If you have specified the /NOLOGOUT qualifier, the process remains connected to the virtual terminal. If the process is executing an image, it continues until the process needs terminal input or attempts to write to the terminal. At that point, the process waits until the physical terminal is reconnected to the virtual terminal.

You can connect to a virtual terminal even if you are not currently using a virtual terminal. However, to log out of your current process you must use the CONNECT command with the /LOGOUT qualifier. If you connect to a virtual terminal from another virtual terminal, you can save your current process by using the /NOLOGOUT qualifier.

Qualifiers

/CONTINUE
/NOCONTINUE (default)
Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This qualifier allows an interrupted image to continue processing after you connect to another process.

The /CONTINUE qualifier is incompatible with the /LOGOUT qualifier.

/LOGOUT (default)
/NOLOGOUT
Logs out your current process when you connect to another process using a virtual terminal.

When you enter the CONNECT command from a process that is not connected to a virtual terminal, you must specify the /LOGOUT qualifier. Otherwise, DCL displays an error message.

The /LOGOUT qualifier is incompatible with the /CONTINUE qualifier.

Examples

1.
```
$ RUN AVERAGE
Ctrl/Y
$ CONNECT/CONTINUE VTA72
```

   In this example, you use the RUN command to execute the image AVERAGE.EXE. You enter this command from a terminal that is connected to a virtual terminal. Next, you press Ctrl/Y to interrupt the image. After you interrupt the image, enter the CONNECT command with the /CONTINUE qualifier. This operation issues the CONTINUE command, so the image continues to run and connects you to another virtual terminal. You can reconnect to the process later.

2.
```
$ SHOW USERS/FULL
     VAX/VMS User Processes at 22-DEC-1994 14:11:56.91
   Total number of users = 51,  number of processes = 158

Username    Node   Process Name       PID     Terminal
KIDDER     BUKETT  KIDDER           29A0015E  FTA3:
KIDDER     BUKETT  _FTA4:           29A0015F  FTA4:
KIDDER     RACEY1  KIDDER           05800062  FTA5:
KIDDER     RACEY1  DECW$MWM         0580005D  MBA44:   Disconnected
KIDDER     RACEY1  DECW$SESSION     05800059
KIDDER     RACEY1  VUE$KIDDER_2     0580005E  (subprocess of 05800059)
KIDDER     RACEY1  VUE$KIDDER_3     0580005F  MBA51:   Disconnected
KIDDER     RACEY1  VUE$KIDDER_4     05800060  MBA53:   Disconnected
SMITH      BUKETT  SMITH            29A002C1  FTA7:
SMITH      BUKETT  SMITH_1          29A006C2  (subprocess of 29A002C1)
SMITH      BUKETT  SMITH_2          29A00244  (subprocess of 29A002C1)
SMITH      HAMLET  SMITH            24800126  FTA6:
SMITH      HAMLET  DECW$BANNER      24800155  (subprocess of 24800126)
SMITH      HAMLET  DECW$MWM         2480011F  MBA170:  Disconnected
SMITH      HAMLET  DECW$SESSION     2480011D  FTA5:
.
.
.
$ CONNECT VTA273
  SMITH         logged out at 22-DEC-1994 14:12:04.53
$
```

This example shows how to reconnect to your original process after you have lost the carrier signal. First, you must log in again and create a new process. After you log in, enter the SHOW USERS/FULL command to determine the virtual terminal name for your initial process. Then enter the CONNECT command to connect to the virtual terminal associated with your original process. The process from which you enter the CONNECT command is logged out because you have not specified any qualifiers.

When you reconnect to the original process, you continue running the image that you were running when you lost the carrier signal. In this example, the user SMITH was at interactive level when the connection was broken.

CONTINUE

Resumes execution of a DCL command, a program, or a command procedure that was interrupted by pressing Ctrl/Y or Ctrl/C. You cannot resume execution of the image if you have entered a command that executes another image or if you have invoked a command procedure.

Format

CONTINUE

Parameters

None.

Description

The CONTINUE command enables you to resume processing an image or a command procedure that was interrupted by pressing Ctrl/Y or Ctrl/C. You cannot resume execution of the image if you have entered a command that executes another image or if you have invoked a command procedure. However, you can use CONTINUE after commands that do not execute separate images; for a list of these commands, see the *OpenVMS User's Manual*.

You can abbreviate the CONTINUE command to a single letter, C.

The CONTINUE command serves as the target command of an IF or ON command in a command procedure. The CONTINUE command is also a target command when it follows a label that is the target of a GOTO command. In addition, you can use the CONTINUE command to resume processing of a program that has executed either a VAX FORTRAN PAUSE statement or a VAX COBOL-74 STOP literal statement.

Examples

1.
```
$ RUN  MYPROGRAM_A
 Ctrl/Y
$ SHOW TIME
  14-DEC-1994 13:40:12
$ CONTINUE
```

   In this example, the RUN command executes the program MYPROGRAM_A. While the program is running, pressing Ctrl/Y interrupts the image. The SHOW TIME command requests a display of the current date and time. The CONTINUE command resumes the image.

2.
```
$ ON SEVERE_ERROR THEN CONTINUE
```

   In this example, the command procedure statement requests the command interpreter to continue executing the procedure if any warning, error, or severe error status value is returned from the execution of a command or program. This ON statement overrides the default action, which is to exit from a procedure following errors or severe errors.

CONVERT

Invokes the Convert utility, which copies records from one file to another and changes the organization and format of the input file to those of the output file. For a complete description of the Convert utility, see the *OpenVMS Record Management Utilities Reference Manual*.

Format

CONVERT input-filespec[,...] output-filespec

CONVERT/DOCUMENT

Converts a CDA supported revisable input file to another revisable or final form output file.

Note

You can use this command only if DECwindows Motif for OpenVMS is installed on your system.

Format

CONVERT/DOCUMENT input-filespec output-filespec

Parameters

input-filespec
Specifies the name of the input file to be converted. The default file type is .DDIF.

output-filespec
Specifies the name of the output file. The default file type is .DDIF.

Description

The CONVERT/DOCUMENT command lets you convert documents from one format to another. You specify the name and format of the input file (a file whose format is incompatible with the application that needs to read the file) and the output file (the file to be created in a new format).

You can convert a file from one format to another if an input converter exists for the input file format and an output converter exists for the output file format. The default input and output file format is DDIF (DIGITAL Document Interchange Format). DDIF is a standard format for the storage and interchange of compound documents, which can include text, graphics, and images.

DDIF input and output converters, in addition to several other converters, are installed with the CDA Base Services for DECwindows Motif for OpenVMS. Some of the converters support processing options, which ensure minimal changes when your input file is converted to a different output file format. Create an options file with the processing options you need before specifying the ev CONVERT /DOCUMENT command with the /OPTIONS qualifier.

Every converter supports a message log option, which is a file name you specify and to which informational and error messages are logged during the conversion.

Qualifiers

/FORMAT=format-name
Specifies the encoding format of the input or output file. The default input and output format is DDIF.

Input converters bundled with the CDA Base Services for DECwindows Motif for OpenVMS and the default file type for the file formats they support are as follows:

Input Format	File Type
DDIF	.DDIF
DTIF	.DTIF
TEXT	.TXT

Output converters bundled with the CDA Base Services for DECwindows Motif for OpenVMS and the default file types for the file formats they support are as follows:

Output Format	File Type
DDIF	.DDIF
DTIF	.DTIF
TEXT	.TXT
PS	.PS
ANALYSIS	.CDA$ANALYSIS

Digital's CDA Converter Library is a layered product that offers several other document, graphics, image, and data table input and output converters. Independent software vendors also write CDA conforming applications and converters for the operating system. Contact your system manager for a complete list of converters available on your system.

Analysis Output Converter

The Analysis output converter produces an analysis of the intermediate representation of the input file. The analysis output file shows the named objects and values stored in the input file. Application programmers use an analysis output file for debugging purposes.

Application end users use an analysis output file to determine whether an input file contains references or links to multiple subfiles. Each subfile must be copied separately across a network because subfiles are not automatically included when an input file is transferred across the network.

You can search the analysis output file for all occurrences of the string ''ERF_''. The following example shows that the image file ''griffin.img'' is linked to the DDIF compound document that is the input file:

```
ERF_LABEL ISO LATIN1 "griffin.img" ! Char. string.
ERF_LABEL TYPE RMS_LABEL TYPE "$RMS:
ERF_CONTROL COPY_REFERENCE ! Integer = 1
```

Note that an analysis output file is intended as a programmer's tool. The coded information in the file is not intended for modification but rather to examine the content of a file. The previous example shows how you can search analysis output for references to linked files.

DDIF Input Converter

The DDIF input converter converts a DDIF input file to an intermediate representation that is subsequently converted to the specified output file format. The following list summarizes the data mapping, conversion restrictions, external file references, and document syntax errors relevant to the DDIF input converter.

- Data mapping

  The information in the DDIF input file maps directly to an intermediate representation.

- Conversion restrictions

  The DDIF input file does not lose any information when converted to the intermediate representation.

  However, if the DDIF input file is a newer version of the DDIF grammar than that understood by the DDIF input converter, data represented by the new grammar elements is lost.

- External file references

  Any external file references within the DDIF input file are converted to the intermediate representation.

  The DDIF input converter makes no attempt to resolve external references, although the converter kernel can if requested by the output converter.

- Document syntax errors

  A document syntax error in the DDIF input file causes a fatal input processing error. If the DDIF input converter encounters a document syntax error, the conversion stops and no further input processing occurs.

DDIF Output Converter

The DDIF output converter creates a DDIF output file from the intermediate representation of the input file. The following list summarizes the data mapping and conversion restrictions relevant to the DDIF output converter.

- Data mapping

  The information in the intermediate representation of the input file maps directly to the DDIF output file.

- Conversion restrictions

  The intermediate representation of the input file does not lose any information when converted to the DDIF output file.

DTIF Input Converter

The DTIF input converter converts a DTIF input file to an intermediate representation that is subsequently converted to the specified output file format. The following list summarizes the data mapping, conversion restrictions, external file references, and document syntax errors relevant to the DTIF input converter.

- Data mapping

  The information in the DTIF input file maps directly to an intermediate representation.

- Conversion restrictions

  The DTIF input file does not lose any information when converted to the intermediate representation.

  However, if the DTIF input file is a newer version of the DTIF grammar than that understood by the DTIF front end, data represented by the new grammar elements is lost.

- External file references

  Any external file references within the DTIF input file are converted to the intermediate representation.

The DTIF input converter makes no attempt to resolve external references.

- Document syntax errors

  A document syntax error in the DTIF input file causes a fatal input processing error. If the DTIF input converter encounters a document syntax error, the conversion stops and no further input processing occurs.

DTIF Output Converter

The DTIF output converter converts the intermediate representation of the input file to a DTIF output file. The following list summarizes the data mapping, conversion restrictions, and external file references relevant to the DTIF output converter.

- Data mapping

  The information in the intermediate representation of the input file maps directly to the DTIF output file.

- Conversion restrictions

  The intermediate representation of the input file does not lose any information when converted to the DTIF output file.

- External file references

  The DTIF output converter converts external file references stored in the intermediate representation of the input file but makes no attempt to resolve external references.

Text Input Converter

The Text input converter converts a Text (ISO Latin1) input file to an intermediate representation that is subsequently converted to the specified output file format. The following list summarizes the data mapping, conversion restrictions, external file references, and document syntax errors relevant to the Text input converter:

- Data mapping

  The information in the text input file maps directly to an intermediate representation. Line breaks and form feeds are mapped to DDIF directives. One or more contiguous blank lines are interpreted as end-of-paragraph markers.

  If the text input file was entered as a DEC Multinational character set file on a character-cell terminal or terminal emulator, the following conversions occur:

Original Character	Converted Character
Concurrency sign	Diaeresis
Capital OE ligature	Multiplication sign
Capital Y with diaeresis	Capital Y with acute accent
Small oe ligature	Division sign
Small y with diaeresis	Y with acute accent

- Conversion restrictions

  The text input file does not lose any information when converted to the intermediate representation because no structure information is contained in a text file.

All nonprinting characters are converted to space characters. For example, characters introducing ANSI escape characters are converted to space characters. There is no attempt to interpret ANSI escape sequences.

- External file references

  Text files do not contain external file references.

- Document syntax errors

  Text files do not contain syntax, so syntax errors are not reported by the Text input converter.

Text Output Converter

The Text output converter converts the intermediate representation of the input file to a Text output file. The following list summarizes the data mapping and conversion restrictions relevant to the Text output converter.

- Data mapping

  All Latin1 text in the intermediate representation of the input file is converted to the text output file.

  When converting an input file to a text output file, you should be aware that text output files can contain only textual content and minimal formatting such as line feeds, page breaks, and tabs. The Text output converter preserves formatting information to the extent possible. Page coordinates convert to the nearest character cell (line,column) position.

- Conversion restrictions

  All graphics, images, and text attributes in the intermediate representation of the input file are lost when converted to the text output file.

  Because a monospace font is used, it is possible some text may be lost due to overwriting to preserve the layout. It is also possible that lines can be truncated if the specified page width is smaller than the page width specified in the document's format information. Neither of these cases occur when you use the OVERRIDE_FORMAT processing option because, in that case, the document's format information is ignored.

PostScript Output Converter

The PostScript output converter converts the intermediate representation of the input file to a PostScript output file. The following list summarizes the data mapping and conversion restrictions relevant to the PostScript output converter.

- Data mapping

  The information in the intermediate representation of the input file maps directly to the PostScript output file.

- Conversion restrictions

  The intermediate representation of the input file does not lose any information when converted to the PostScript output file.

/MESSAGE_FILE=*filespec*
/NOMESSAGE_FILE (default)

Turns on message logging for document conversion. Messages output by the input and output converters are directed to the file specified with *filespec*. If filespec is not specified, messages are output to SYS$ERROR. The default is /NOMESSAGE_FILE.

/OPTIONS=*options-filename*

Specifies a text file that contains processing options applied to the input file and the output file during the conversion. The default file type for an options file is .CDA$OPTIONS.

Creating the Options File

You can create an options file prior to specifying the CONVERT/DOCUMENT command with the /OPTIONS qualifier. An options file is a text file with a default file type of .CDA$OPTIONS on the operating system.

The options file contains all the processing options for your input file format and your output file format. Processing options help ensure minimal changes when your input file is converted to a different output file format.

An options file is not required. Default processing options are applied automatically when you convert a file. However, you may require an options file if you need to use other than the default settings.

Use the following guidelines to create an options file:

- Begin each line of the options file with the keyword for the input or output format, followed by one or more spaces or tabs, or by a slash (/).

  For some file formats, such as DDIF and DTIF, there is an input converter and an output converter. You can restrict a processing option to only the input format or the output format by following the format keyword with _INPUT or _OUTPUT.

- Specify only one processing option on each line when there are several options for the same input or output format.

- Use uppercase and lowercase alphabetic characters, digits (0-9), dollar signs ($), and underscores (_) to specify the processing options.

- Use one or more spaces or tabs to precede values specified for a processing option.

The following example is a typical entry in an options file:

```
PS PAPER_HEIGHT 10
```

In this example, the extension _OUTPUT is not required for the format keyword, since PostScript is available only as an output format. The value specified for PAPER_HEIGHT is in inches by default.

If the options file includes options that do not apply to the converters for a particular conversion, those options are ignored.

If you specify an invalid option for an input or output format or an invalid value for an option, you receive an error message. The processing options described in the following sections document any restrictions.

Processing Options for Analysis Output

The Analysis output converter supports the following options:

- *COMMENT DEFAULT_VALUES*

  Inserts a comment character (!) at the beginning of lines generated by default values. (The comment prefix is also included on associated aggregate brackets and array parentheses where they may apply.)

- *COMMENT INHERITED_VALUES*

  Inserts a comment character (!) at the beginning of lines generated by inherited values. (The comment prefix is also included on associated aggregate brackets and array parentheses where they may apply.)

- *TRANSLATE_BYTE_STRINGS*

  Overrides the default. For data of type BYTE STRING, the analysis output no longer displays the hexadecimal translation if all the characters in the byte string are printable characters (hex values 20 through 7E). This feature may be overridden by supplying the TRANSLATE_BYTE_STRINGS option.

- *IMAGE_DATA*

  Overrides the default. For the special case of byte string data for item DDIF$_IDU_PLANE_DATA (a bitmapped image), the analysis output previously included both a hexadecimal and an ASCII translation display, neither of which were of particular value to most users. With the new version, both displays will be replaced with the following comment:

  ```
  ! *** Bit-mapped data not displayed here ***
  ```

  To retain the hexadecimal display, supply the IMAGE_DATA option. Even with this option turned on, there will be no translation into ASCII.

- *INHERITANCE*

  Specifies that the analysis is shown with attribute inheritance enabled. Inherited attributes are marked as ‘‘[Inherited value.]’’ in the output. This option also causes external references to be imported into the main document.

Processing Options for Text Output

The Text output converter supports the following options:

- *ASCII_FALLBACK [ON,OFF]*

  Causes the Text output converter to output text in 7-bit ASCII. The fallback representation of the characters is described in the ASCII standard. If this option is not specified, the default is OFF; if this option is specified without a value, the default is ON.

- *CONTENT_MESSAGES [ON,OFF]*

  Causes the Text output converter to put a message in the output file each time a nontext element is encountered in the intermediate representation of the input file. If this option is not specified, the default is OFF; if this option is specified without a value, the default is ON.

- *HEIGHT value*

  Specifies the maximum number of lines per page in your text output file. If you specify zero, the number of lines per page will correspond to the height specified in your document. If you also specify OVERRIDE_FORMAT, or if the document has no inherent page size, the document is formatted to the height value specified by this option. The default height is 66 lines.

- *OVERRIDE_FORMAT [ON,OFF]*

  Causes the Text output converter to ignore the document formatting information included in your document, so that the text is formatted in a single large galley per page that corresponds to the size of the page as specified by the HEIGHT and WIDTH processing options. If this option is not specified, the default is OFF; if this option is specified without a value, the default is ON.

- *SOFT_DIRECTIVES [ON,OFF]*

  Causes the Text output converter to obey the soft directives contained in the document when creating your text output file. If this option is not specified, the default is OFF; if this option is specified without a value, the default is ON.

- *WIDTH value*

  Specifies the maximum number of columns of characters per page in your text output file. If you specify zero, the number of columns per page will correspond to the width specified in your document. If you also specify OVERRIDE_FORMAT, or if the document has no inherent page size, the document is formatted to the value specified by this processing option. If any lines of text exceed this width value, the additional columns are truncated. The default width is 80 characters.

PostScript Output Converter

The PostScript output converter supports the following options:

- *PAPER_SIZE size*

  Specifies the size of the paper to be used when formatting the resulting PostScript output file. Valid values for the *size* argument are as follows:

Keyword	Size
A0	841 x 1189 millimeters (33.13 x 46.85 inches)
A1	594 x 841 millimeters (23.40 x 33.13 inches)
A2	420 x 594 millimeters (16.55 x 23.40 inches)
A3	297 x 420 millimeters (11.70 x 16.55 inches)
A4	210 x 297 millimeters (8.27 x 11.70 inches)
A	8.5 x 11 inches
B	11 x 17 inches
C	17 x 22 inches
D	22 x 34 inches
E	34 x 44 inches
LEDGER	11 x 17 inches
LEGAL	8.5 x 14 inches
LETTER	8.5 x 11 inches
LP	13.7 x 11 inches
VT	8 x 5 inches

  The A paper size (8.5 x 11 inches) is the default.

- *PAPER_HEIGHT height*

  Specifies a paper size other than one of the predefined values provided. The default paper height is 11 inches.

- *PAPER_WIDTH width*

  Specifies a paper size other than one of the predefined sizes provided. The default paper width is 8.5 inches.

- *PAPER_TOP_MARGIN top-margin*

  Specifies the width of the margin provided at the top of the page. The default value is 0.25 inch.

- *PAPER_BOTTOM_MARGIN bottom-margin*

  Specifies the width of the margin provided at the bottom of the page. The default value is 0.25 inch.

- *PAPER_LEFT_MARGIN left-margin*

  Specifies the width of the margin provided on the left-hand side of the page. The default value is 0.25 inch.

- *PAPER_RIGHT_MARGIN right-margin*

  Specifies the width of the margin provided on the right-hand side of the page. The default value is 0.25 inch.

- *PAPER_ORIENTATION orientation*

  Specifies the paper orientation to be used in the output PostScript file. The valid values for the *orientation* argument are as follows:

Keyword	Meaning
PORTRAIT	The page is oriented so that the larger dimension is parallel to the vertical axis.
LANDSCAPE	The page is oriented so that the larger dimension is parallel to the horizontal axis.

  The default is PORTRAIT.

- *EIGHT_BIT_OUTPUT [ON,OFF]*

  Specifies whether the PostScript output converter should use 8-bit output. The default value is ON.

- *LAYOUT [ON,OFF]*

  Specifies whether the PostScript output converter processes the layout specified in the DDIF document. The default value is ON.

- *OUTPUT_BUFFER_SIZE output-buffer-size*

  Specifies the size of the output buffer. The value you specify must be within the range 64 to 256. The default value is 132.

- *PAGE_WRAP [ON,OFF]*

  Specifies whether the PostScript output converter performs page wrapping of any text that would exceed the bottom margin. The default value is ON.

- *SOFT_DIRECTIVES [ON,OFF]*

  Specifies whether the PostScript output converter processes soft directives in the DDIF file in order to format output. (Soft directives specify such formatting commands as new line, new page, and tab.) If the PostScript output converter processes soft directives, the output file will look more like you intended. The default value is ON.

- *WORD_WRAP [ON,OFF]*

  Specifies whether the PostScript output converter performs word wrapping of any text that would exceed the right margin. The default value is ON. If you specify OFF, the PostScript output converter allows text to exceed the right margin.

Domain Converter

You might create an options file containing processing options that apply to any CDA supported tabular file format for which there is an input converter. Data tables and spreadsheets are examples of tabular file formats.

To convert tabular input files to document output files, use the DTIF_TO_DDIF format name, followed by the processing options described in this section. Specify the DTIF_TO_DDIF processing options in addition to the processing options for a particular tabular input file format and a particular document output file format.

You might want to convert tabular input files to document output files so that you can include textual representations of tables in reports and other documents. You should be aware, however, that you lose cell borders, headers, grid lines, all formulas, and font types when converting a tabular input file to a document output file.

The domain converter supports the following options:

- *COLUMN_TITLE*

  Displays the column titles as contained in the column attributes centered at the top of the column.

- *CURRENT_DATE*

  Displays the current date and time in the bottom left corner of the page. The value is formatted according to the document's specification for a default date and time.

- *DOCUMENT_DATE*

  Displays the document date and time as contained in the document header in the top left corner of the page. The value is formatted according to the document's specification for a default date and time.

- *DOCUMENT_TITLE*

  Displays the document title or titles as contained in the document header centered at the top of the page, one string per line.

- *PAGE_NUMBER*

  Displays the current page number in the top right corner of the page.

- *PAPER_SIZE size*

  Specifies the size of the paper to be used when formatting the resulting PostScript output file. Valid values for the size argument are as follows:

Keyword	Size
A0	841 x 1189 millimeters (33.13 x 46.85 inches)
A1	594 x 841 millimeters (23.40 x 33.13 inches)
A2	420 x 594 millimeters (16.55 x 23.40 inches)

Keyword	Size
A3	297 x 420 millimeters (11.70 x 16.55 inches)
A4	210 x 297 millimeters (8.27 x 11.70 inches)
A5	148 x 210 millimeters (5.83 x 8.27 inches)
A	8.5 x 11 inches (216 x 279 millimeters)
B	11 x 17 inches (279 x 432 millimeters)
B4	250 x 353 millimeters (9.84 x 13.90 inches)
B5	176 x 250 millimeters (6.93 x 9.84 inches)
C	17 x 22 inches (432 x 559 millimeters)
C4	229 x 324 millimeters (9.01 x 12.76 inches)
C5	162 x 229 millimeters (6.38 x 9.02 inches)
D	22 x 34 inches (559 x 864 millimeters)
DL	110 x 220 millimeters (4.33 x 8.66 inches)
E	34 x 44 inches (864 x 1118 millimeters)
10x13_ ENVELOPE	13 x 254 millimeters (15600 x 10 inches)
9x12_ ENVELOPE	12 x 229 millimeters (14400 x 9 inches)
BUSINESS_ ENVELOPE	9.5 x 105 millimeters (11400 x 4.13 inches)
EXECUTIVE	10 x 191 millimeters (12000 x 7.5 inches)
LEDGER	11 x 17 inches (279 x 432 millimeters)
LEGAL	8.5 x 14 inches (216 x 356 millimeters)
LETTER	8.5 x 11 inches (216 x 279 millimeters)
LP	13.7 x 11 inches (348 x 279 millimeters)
VT	8 x 5 inches (203 x 127 millimeters)

The A paper size (8.5 x 11 inches) is the default.

- *PAPER_HEIGHT height*

  Specifies a paper size other than one of the predefined values provided. The default paper height is 11 inches.

- *PAPER_WIDTH width*

  Specifies a paper size other than one of the predefined sizes provided. The default paper width is 8.5 inches.

- *PAPER_TOP_MARGIN top-margin*

  Specifies the width of the margin provided at the top of the page. The default value is 0.25 inch.

- *PAPER_BOTTOM_MARGIN bottom-margin*

  Specifies the width of the margin provided at the bottom of the page. The default value is 0.25 inch.

- *PAPER_LEFT_MARGIN left-margin*

  Specifies the width of the margin provided on the left side of the page. The default value is 0.25 inch.

- *PAPER_RIGHT_MARGIN right-margin*

  Specifies the width of the margin provided on the right side of the page. The default value is 0.25 inch.

- *PAPER_ORIENTATION orientation*

  Specifies the paper orientation to be used in the output file. The valid values for the *orientation* argument are as follows:

Keyword	Meaning
PORTRAIT	The page is oriented so that the larger dimension is parallel to the vertical axis.
LANDSCAPE	The page is oriented so that the larger dimension is parallel to the horizontal axis.

  The default is PORTRAIT.

Example

```
$ CONVERT/DOCUMENT/OPTIONS=MY_OPTIONS.CDA$OPTIONS -
_$MY_INPUT.DTIF/FORMAT=DTIF MY_OUTPUT.DDIF/FORMAT=DDIF
```

This command converts an input file named MY_INPUT.DTIF, which has the DTIF format, to an output file named MY_OUTPUT.DDIF, which has the DDIF format. The specified options file is named MY_OPTIONS.CDA$OPTIONS.

CONVERT/RECLAIM

Invokes the Convert/Reclaim utility, which makes empty buckets in Prolog 3 indexed files available so that new records can be written in them. The /RECLAIM qualifier is required. For a complete description of the Convert /Reclaim utility, see the *OpenVMS Record Management Utilities Reference Manual*.

Format

CONVERT/RECLAIM filespec

COPY

Creates a new file from one or more existing files. The COPY command can do the following:

- Copy an input file to an output file.
- Concatenate two or more input files into a single output file.
- Copy a group of input files to a group of output files.

Format

COPY input-filespec[,...] output-filespec

Parameters

input-filespec[,...]

Specifies the name of an existing file to be copied. The asterisk (*) and the percent sign (%) wildcard characters are allowed. If you do not specify the device or directory, the COPY command uses your current default device and directory. If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+).

output-filespec

Specifies the name of the output file into which the input is copied.

You must specify at least one field in the output file specification. If you do not specify the device or directory, the COPY command uses your current default device and directory. The COPY command replaces any other missing fields (file name, file type, version number) with the corresponding field of the input file specification. If you specify more than one input file, the COPY command generally uses the fields from the first input file to determine any missing fields in the output file.

You can use the asterisk (*) wildcard character in place of any two of the following: the file name, the file type, or the version number. The COPY command uses the corresponding field in the related input file to name the output file.

Description

The COPY command creates a new file from one or more existing files. If you do not specify the device or directory, the COPY command uses your current default device and directory. The COPY command can do the following:

- Copy an input file to an output file.
- Concatenate two or more input files into a single output file.
- Copy a group of input files to a group of output files.

The COPY command, by default, creates a single output file. When you specify more than one input file, the first input file is copied to the output file, and each subsequent input file is appended to the end of the output file. If a field of the output file specification is missing or contains an asterisk (*) wildcard character, the COPY command uses the corresponding field from the first, or only, input file to name the output file.

If you specify multiple input files with maximum record lengths, the COPY command gives the output file the maximum record length of the first input file. If the COPY command encounters a record in a subsequent input file that is longer than the maximum record length of the output file, it issues a message noting the incompatible file attributes and begins copying the next file.

To create multiple output files, specify multiple input files and use at least one of the following:

- An asterisk (*) wildcard character in the output directory specification, file name, file type, or version number field
- Only a node name, a device name, or a directory specification as the output file specification
- The /NOCONCATENATE qualifier

When the COPY command creates multiple output files, it uses the corresponding field from each input file in the output file name. You also can use the asterisk (*) wildcard character in the output file specification to have COPY create more than one output file. For example:

```
$ COPY A.A;1, B.B;1 *.C
```

This COPY command creates the files A.C;1 and B.C;1 in the current default directory. When you specify multiple input and output files you can use the /LOG qualifier to verify that the files were copied as you intended.

Note that there are special considerations for using the COPY command with DECwindows compound documents. For more information, see the *Guide to OpenVMS File Applications*.

Version Numbers

If you do not specify version numbers for input and output files, the COPY command (by default) assigns a version number to the output files that is either of the following:

- The version number of the input file
- A version number one greater than the highest version number of an existing file with the same file name and file type

When you specify the output file version number by an asterisk (*) wildcard character, the COPY command uses the version numbers of the associated input files as the version numbers of the output files.

If you specify the output file version number by an explicit version number, the COPY command uses that number for the output file specification. If a higher version of the output file exists, the COPY command issues a warning message and copies the file. If an equal version of the output file exists, the COPY command issues a message and does *not* copy the input file.

File Protection and Creation/Revision Dates

The COPY command considers an output file to be new when you specify any portion of the output file name explicitly. The COPY command sets the creation date for a new file to the current time and date.

If you specify the output file by one or more asterisk (*) and percent sign (%) wildcard characters, the COPY command uses the creation date of the input file.

The COPY command always sets the revision date of the output file to the current time and date; it sets the backup date to zero. The file system assigns the output file a new expiration date. (The file system sets expiration dates if retention is enabled; otherwise, it sets expiration dates to zero.)

The protection and access control list (ACL) of the output file is determined by the following parameters, in the following order:

- Protection of previously existing versions of the output file
- Default Protection and ACL of the output directory
- Process default file protection

(Note that the BACKUP command takes the creation and revision dates as well as the file protection from the input file.)

Use the /PROTECTION qualifier to change the output file protection.

Normally, the owner of the output file will be the same as the creator of the output file. However, if a user with extended privileges creates the output file, the owner will be the owner of the parent directory or of a previous version of the output file if one exists.

Extended privileges include any of the following:

- SYSPRV (system privilege) or BYPASS
- System user identification code (UIC)
- GRPPRV (group privilege) if the owner of the parent directory (or previous version of the output file) is in the same group as the creator of the new output file
- An identifier (with the resource attribute) representing the owner of the parent directory (or the previous version of the output file)

Copying Directory Files

If you copy a file that is a directory, the COPY command creates a new *empty* directory of the named directory. The COPY command does *not* copy any files from the named directory to the new directory. See the examples section for examples of copying directory files.

Qualifiers

/ALLOCATION=number-of-blocks

Forces the initial allocation of the output file to the specified number of 512-byte blocks. If you do not specify the /ALLOCATION qualifier, or if you specify it without the *number-of-blocks* parameter, the initial allocation of the output file is determined by the size of the input file being copied.

/BACKUP

Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /BACKUP qualifier selects files according to the dates of their most recent backups. This qualifier is incompatible with the /CREATED, /EXPIRED, and

/MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/BEFORE[=time]
Selects only those files dated prior to the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /BEFORE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/BY_OWNER[=uic]
Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

Specify the UIC by using standard UIC format as described in the *OpenVMS Guide to System Security*.

/CONCATENATE (default)
/NOCONCATENATE
Creates one output file from multiple input files when you do not use the asterisk (*) or percent sign (%) wildcard characters in the output file specification. The /NOCONCATENATE qualifier generates multiple output files. A wildcard character in an input file specification results in a single output file consisting of the concatenation of all input files matching the file specification.

Files from Files–11 On-Disk Structure Level 2 disks are concatenated in alphanumeric order; if you specify an asterisk (*) or percent sign (%) wildcard character in the file version field, files are copied in descending order by version number. Files from Files–11 On-Disk Structure Level 1 disks are concatenated in random order.

/CONFIRM
/NOCONFIRM (default)
Controls whether a request is issued before each copy operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	Ctrl/Z
1	0	ALL
	Return	

You can use any combination of uppercase and lowercase letters for word responses. You can abbreviate word responses to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers include: NO, FALSE, 0, and pressing the Return key. Entering QUIT or pressing Ctrl/Z indicates that you want to stop processing the command at that point. When you respond by entering ALL, the command continues to process but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CONTIGUOUS
/NOCONTIGUOUS
Specifies that the output file must occupy contiguous physical disk blocks. By default, the COPY command creates an output file in the same format as the corresponding input file. Also, by default, if not enough space exists for a contiguous allocation, the COPY command does not report an error. If you copy multiple input files of different formats, the output file may or may not be contiguous. You can use the /CONTIGUOUS qualifier to ensure that files are copied contiguously.

The /CONTIGUOUS qualifier has no effect when you copy files to or from tapes because the size of the file on tape cannot be determined until after it is copied to the disk. If you copy a file from a tape and want the file to be contiguous, use the COPY command twice: once to copy the file from the tape, and a second time to create a contiguous file.

/CREATED (default)
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the /BACKUP, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/EXCLUDE=(filespec[,...])
Excludes the specified files from the copy operation. You can include a directory but not a device in the file specification. The asterisk (*) and the percent sign (%) wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /EXPIRED qualifier selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the /BACKUP, /CREATED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/EXTENSION=n
Specifies the number of blocks to be added to the output file each time the file is extended. If you do not specify the /EXTENSION qualifier, the extension attribute of the corresponding input file determines the default extension attribute of the output file.

/LOG
/NOLOG (default)
Controls whether the COPY command displays the file specifications of each file copied.

When you use the /LOG qualifier, the COPY command displays the following for each copy operation:

- The file specifications of the input and output files
- The number of blocks or the number of records copied (depending on whether the file is copied on a block-by-block or record-by-record basis)

- The total number of new files created

/MODIFIED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /MODIFIED qualifier selects files according to the dates on which they were last modified. This qualifier is incompatible with the /BACKUP, /CREATED, and /EXPIRED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time modifiers, the default is the /CREATED qualifier.

/OVERLAY
/NOOVERLAY (default)
Requests that data in the input file be copied into the existing specified file, overlaying the existing data, rather than allocating new space for the file. The physical location of the file on disk does not change. However, for RMS indexed and relative files, if the output file has fewer blocks allocated than the input file, the copy fails giving an RMS-E-EOF error.

The /OVERLAY qualifier is ignored if the output file is written to a non-file-structured device.

/PROTECTION=(ownership[:access][,...])
Specifies protection for the output file.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), write (W), execute (E), or delete (D).

The default protection, including any protection attributes not specified, is that of the existing output file. If no output file exists, the current default protection applies.

For more information on specifying protection codes, see the *OpenVMS Guide to System Security*.

/READ_CHECK
/NOREAD_CHECK (default)
Reads each record in the input files twice to verify that it has been read correctly.

/REPLACE
/NOREPLACE (default)
Requests that, if a file exists with the same file specification as that entered for the output file, the existing file is to be deleted. The COPY command allocates new space for the output file. In general, when you use the /REPLACE qualifier, include version numbers with the file specifications. By default, the COPY command creates a new version of a file if a file with that specification exists, incrementing the version number. The /NOREPLACE qualifier signals an error when a conflict in version numbers occurs.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as absolute time, as combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /SINCE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/TRUNCATE (default)
/NOTRUNCATE
Controls whether the COPY command truncates an output file at the end-of-file (EOF) when copying it. This operation can only be used with sequential files.

By default, the actual size of the input file determines the size of the output file. If you select /NOTRUNCATE, the allocation of the input file determines the size of the output file.

/VOLUME=n
Places the output file on the specified relative volume number of a multivolume set. By default, the COPY command places the output file arbitrarily in a multivolume set.

/WRITE_CHECK
/NOWRITE_CHECK (default)
Reads each record in the output file after it was written to verify that the record was copied successfully and that the file can be read subsequently without error.

Note

Some hardware devices, such as TK50 tape drives, verify data integrity as part of their hardware function. For devices such as these, you do not need to use /WRITE_CHECK. For information about which devices provide automatic write checking, consult your hardware documentation.

Examples

1.
```
$ COPY TEST.DAT NEWTEST.DAT
```

In this example, the COPY command copies the contents of the file TEST.DAT from the default disk and directory to a file named NEWTEST.DAT on the same disk and directory. If a file named NEWTEST.DAT exists, the COPY command creates a new version of the file.

2.
```
$ COPY ALPHA.TXT TMP
$ COPY ALPHA.TXT .TMP
```

In this example, the first COPY command copies the file ALPHA.TXT into a file named TMP.TXT. The COPY command uses the file type of the input file to complete the file specification for the output file. The second COPY command creates a file named ALPHA.TMP. The COPY command uses the file name of the input file to name the output file.

3.
```
$ COPY/LOG TEST.DAT  NEW.DAT;1/REPLACE
%COPY-I-REPLACED, DBA0:[MAL]NEW.DAT;1 being replaced
%COPY-S-COPIED, DBA0:[MAL]TEST.DAT;1 copied to DBA0:[MAL]NEW.DAT;1 (1 block)
```

In this example, the /REPLACE qualifier requests that the COPY command replace an existing version of the output file with the new file. The first message from the COPY command indicates that it is replacing an existing file. The version number in the output file must be explicit; otherwise, the COPY command creates a new version of the file NEW.DAT.

4. `$ COPY *.COM [MALCOLM.TESTFILES]`

   In this example, the COPY command copies the highest versions of files in the current default directory with the file type .COM to the subdirectory MALCOLM.TESTFILES.

5.
```
$ COPY/LOG *.TXT *.OLD
%COPY-S-COPIED, DBA0:[MAL]A.TXT;2 copied to DBA0:[MAL]A.OLD;2 (1 block)
%COPY-S-COPIED, DBA0:[MAL]B.TXT;2 copied to DBA0:[MAL]B.OLD;2 (1 block)
%COPY-S-COPIED, DBA0:[MAL]G.TXT;2 copied to DBA0:[MAL]G.OLD;2 (4 blocks)
%COPY-S-NEWFILES, 3 files created
```

   In this example, the COPY command copies the highest versions of files with file types .TXT into new files. Each new file has the same file name as an existing file, but a file type .OLD. The last message from the COPY command indicates the number of new files that have been created.

6.
```
$ COPY/LOG  A.DAT,B.MEM C.*
%COPY-S-COPIED, DBA0:[MAL]A.DAT;5 copied to DBA0:[MAL]C.DAT;11 (1 block)
%COPY-S-COPIED, DBA0:[MAL]B.MEM;2 copied to DBA0:[MAL]C.MEM;24 (58 records)
%COPY-S-NEWFILES, 2 files created
```

   In this example, the two input file specifications are separated with a comma. The asterisk (*) wildcard character in the output file specification indicates that two output files are to be created. For each copy operation, the COPY command uses the file type of the input file to name the output file.

7.
```
$ COPY/LOG *.TXT TXT.SAV
%COPY-S-COPIED, DBA0:[MAL]A.TXT;2 copied to DBA0:[MAL]TXT.SAV;1 (1 block)
%COPY-S-APPENDED, DBA0:[MAL]B.TXT;2 appended to DBA0:[MAL]TXT.SAV;1 (3 records)
%COPY-S-APPENDED, DBA0:[MAL]G.TXT;2 appended to DBA0:[MAL]TXT.SAV;1 (51 records)
%COPY-S-NEWFILES, 1 file created
```

   In this example, the COPY command copies the highest versions of all files with the file type TXT to a single output file named TXT.SAV. After the first input file is copied, the messages from the COPY command indicate that subsequent files are being appended to the output file.

   Note that, if you use the /NOCONCATENATE qualifier in this example, the COPY command creates one TXT.SAV file for each input file. Each TXT.SAV file has a different version number.

8. `$ COPY MASTER.DOC DBA1:[BACKUP]`

   In this example, the COPY command copies the highest version of the file MASTER.DOC to the device DBA1. If no file named MASTER.DOC exists in the directory [BACKUP], the COPY command assigns the version number of the input file to the output file. You must have write (W) access to the directory [BACKUP] on device DBA1 for the command to work.

9. `$ COPY SAMPLE.EXE DALLAS::DISK2:[000,000]SAMPLE.EXE/CONTIGUOUS`

   In this example, the COPY command copies the file SAMPLE.EXE on the local node to a file with the same name at remote node DALLAS. The /CONTIGUOUS qualifier indicates that the output file is to occupy consecutive physical disk blocks. You must have write (W) access to the device DISK2 on remote node DALLAS for the command to work.

10.
```
$ COPY *.* PRTLND::*.*
```

In this example, the COPY command copies all files within the user directory at the local node to the remote node PRTLND. The new files have the same names as the input file. You must have write (W) access to the default directory on remote node PRTLND for the command to work.

11.
```
$ COPY BOSTON::DISK2:TEST.DAT;5
_To: DALLAS"SAM SECReturn"::DISK0:[MODEL.TEST]TEST.DAT/ALLOCATION=50
```

In this example, the COPY command copies the file TEST.DAT;5 on the device DISK2 at node BOSTON to a new file named TEST.DAT at remote node DALLAS. The /ALLOCATION qualifier initially allocates 50 blocks for the new file TEST.DAT at node DALLAS. The access control string SAM SECReturn is used to access the remote directory.

12.
```
$ MOUNT   TAPED1:   VOL025   TAPE:
$ COPY  TAPE:*.*   *
```

In this example, the MOUNT command requests that the volume labeled VOL025 be mounted on the magnetic tape device TAPED1 and assigns the logical name TAPE to the device.

The COPY command uses the logical name TAPE as the input file specification, requesting that all files on the magnetic tape be copied to the current default disk and directory. All the files copied retain their file names and file types.

13.
```
$ ALLOCATE CR:
  _CR1: ALLOCATED
$ COPY CR1: CARDS.DAT
$ DEALLOCATE CR1:
```

In this example, the ALLOCATE command allocates a card reader for exclusive use by the process. The response from the ALLOCATE command indicates the device name of the card reader, CR1.

After the card reader is allocated, you can place a deck of cards in the reader and enter the COPY command, specifying the card reader as the input file. The COPY command reads the cards into the file CARDS.DAT. The end-of-file (EOF) in the card deck must be indicated with an EOF card (12-11-0-1-6-7-8-9 overpunch).

The DEALLOCATE command relinquishes use of the card reader.

14.
```
$ COPY [SMITH]CATS.DIR [JONES]
$ COPY [SMITH.CATS]*.* [JONES.CATS]*.*
```

In this example, the COPY command creates the new empty directory [JONES.CATS] that is registered in the [JONES]CATS.DIR directory file. After the COPY command creates the new [JONES]CATS.DIR directory file, you can copy or create files in the [JONES.CATS] directory.

The second COPY command in this example copies files from the [SMITH.CATS] directory to the [JONES.CATS] directory.

15. `$ COPY [SMITH]CATS.DIR [SMITH]DOGS.DIR`

    In this example, the COPY command creates the new empty directory file, called [SMITH]DOGS.DIR. Use this copy command to create a directory file that has the same attributes as the [SMITH]CATS.DIR file. This command example has the same effect as entering the command:

    ```
    $ CREATE/DIRECTORY [SMITH.DOGS]
    ```

16.
```
$ COPY [SMITH]CATS.DIR [SMITH.ANIMALS]
$ COPY [SMITH.CATS]*.* [SMITH.ANIMALS.CATS]*.*
$ DELETE [SMITH.CATS]*.*;*
$ SET PROTECTION=W=D CATS.DIR
$ DELETE CATS.DIR;
```

    In this example, the COPY command creates the new empty directory file called [SMITH.ANIMALS]CATS.DIR. The subsequent commands in this example then copy the files from the [SMITH.CATS] directory to the [SMITH.ANIMALS.CATS] directory, then delete the original CATS.DIR directory file. Because the CATS.DIR is a directory file, you must specify a protection code of delete (D) before you can delete the directory file.

COPY/FTP

Transfers files between hosts with possibly dissimilar file systems over a TCP/IP connection by invoking the FTP utility.

Format

COPY/FTP fromFile toFile

Parameters

fromFile
Specifies the name of an existing file (the source file) to be copied.

toFile
Specifies the name of the output file (the destination file) into which the input file is copied.

Description

The COPY/FTP command copies files to and from remote nodes using the File Transfer Protocol (FTP). The services provided by this command are a subset of the architected features of FTP (see vendor documentation for usage of their supplied FTP program).

For OpenVMS to OpenVMS Transfers

If both machines support OpenVMS structured transfers, the /BINARY, /ASCII, and /FDL qualifiers will be ignored. The cooperating OpenVMS FTP client and server will automatically transfer the file with proper OpenVMS attributes.

COPY/FTP commonly supports the asterisk wildcard character (*) in remote file specifications.

Qualifiers

/ANONYMOUS
Causes an anonymous access to the remote node or nodes. /ANONYMOUS is the default remote access. The password passed to the remote node should be in the form of "user@fullyqualifiednodename".

/ASCII
Used to identify an ASCII file (text file). /ASCII is the default.

/BINARY
Required to identify binary files.

/FDL
This qualifier is optional. Causes interaction with an FDL (file definition language) file. If the file is being copied to the local OpenVMS system, a remote FDL file is sought and interpreted for the operation. If the file is being copied outside the local OpenVMS system, an FDL file is generated and copied in addition to the requested file. If the /FDL qualifier is specified and the vendor application does not support it, a warning message may be issued.

/LOG
Displays a message at SYS$OUTPUT when a file is transferred.

/NOSTRUVMS
Used to explicitly disable the negotiation of STRU OpenVMS transfers. Otherwise, some servers will immediately abort when negotiating the feature.

/VERBOSE
/NOVERBOSE
Specifies whether all messages (including banner messages) are to be displayed on the terminal. By default, disables the display of the messages.

Examples

1. `$ COPY/FTP/FDL/ANON rms_indexed_file.idx remotehst5::"/public/rms.idx.file"`

   This example transfers the OpenVMS RMS file *rms_indexed_file.idx* to the remote file *public/rms.idx.file* on *remotehst5* over a TCP/IP connection. Access to the remote host is anonymous and an FLD file is generated and copied along with *rms_indexed_file.idx*.

COPY/RCP

Copies files from host to host over a TCP/IP connection by invoking the RCP utility.

Format

COPY/RCP fromFile toFile

Parameters

fromFile
Specifies the name of an existing file (the source file) to be copied.

toFile
Specifies the name of the output file (the destination file) into which the input file is copied.

Description

The COPY/RCP command copies one or more files (or directory trees) to or from a remote host using the RCP utility.

The OpenVMS DCL commands for TCP/IP support the same remote file specification format as the DCL commands for DECnet network connections. Some implementations of the file transaction applications support file transfers in which both the source file and the destination file are remote file specifications.

The full format for a remote file specification is as follows:

node"username password account"::filename.ext

If a file resides on a non OpenVMS system, enclose the name of the file in quotation marks. For example, to access a file named */usr/users/user/Orders* on a DEC OSF/1 node named U32, you would use the following format for the file specification:

```
U32"user password"::"/usr/users/user/Orders"
```

Unlike OpenVMS, UNIX systems support case-sensitive file specifications.

Qualifiers

/AUTHENTICATE
Specifies that Kerberos authentication should be used for acquiring access to the remote node.

/LOG
Displays a message in SYS$OUTPUT when a file is transferred.

/PRESERVE
Preserves the file protection codes.

/RECURSIVE
Requests a subdirectory copy operation.

/TRUNCATE=USERNAME
Truncates the user name to 8 characters.

/USERNAME=username
Optional qualifier that specifies the remote user name. The standard operation is to log in to a remote system using the same user name as at the local terminal. The command supports quoted parameters in the /USERNAME value.

Examples

1. `$ COPY/RCP local_file.c remotehst4"Smith smpw"::rem_file.c`

   This example copies *local_file.c* to *rem_file.c* on the remote host *remotehst4* over a TCP/IP connection.

CREATE

Creates a sequential disk file (or files).

Format

CREATE filespec[,...]

Parameter

filespec[,...]
Specifies the name of one or more input files to be created. Wildcard characters are not allowed. If you omit either the file name or the file type, the CREATE command does not supply any defaults. The file name or file type is null. If the specified file already exists, a new version is created.

Description

The CREATE command creates a new sequential disk file. In interactive mode, each separate line that you enter after you enter the command line becomes a record in the newly created file. To terminate the file input, press Ctrl/Z.

When you enter the CREATE command from a command procedure file, the system reads all subsequent records in the command procedure file into the new file until it encounters a dollar sign ($) in the first position in a record. Terminate the file input with a line with a dollar sign in column 1 (or with the end of the command procedure).

If you use an existing file specification with the CREATE command, the newly created file has a higher version number than any existing files with the same specification.

If you use the CREATE command to create a file in a logical name search list, the file will only be created in the first directory produced by the logical name translation.

Normally, the owner of the output file will be the same as the creator of the output file. However, if a user with extended privileges creates the output file, the owner will be the owner of the parent directory or any previous versions of the output file.

Extended privileges include any of the following:

- SYSPRV (system privilege) or BYPASS
- System user identification codes (UICs)
- GRPPRV (group privilege) if the owner of the parent directory (or previous version of the output file) is in the same group as the creator of the new output file
- An identifier (with the resource attribute) representing the owner of the parent directory (or previous version of the output file)

Qualifiers

/LOG
/NOLOG (default)
Displays the file specification of each new file created as the command executes.

/OWNER_UIC=uic
Requires SYSPRV (system privilege) privilege to specify a user identification code (UIC) other than your own.

Specifies the UIC to be associated with the file being created. Specify the UIC by using standard UIC format as described in the *OpenVMS User's Manual*.

/PROTECTION=(ownership[:access][,...])
Specifies protection for the file.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), write (W), execute (E), or delete (D).

If you do not specify a value for each access category, or if you omit the /PROTECTION qualifier, the CREATE command applies the following protection for each unspecified category:

File Already Exists?	Protection Applied
Yes	Protection of the existing file
No	Current default protection

For more information on specifying protection codes, see the *OpenVMS Guide to System Security*.

/VOLUME=n
Places the file on the specified relative volume of a multivolume set. By default, the file is placed arbitrarily in a multivolume set.

Examples

1.
```
$ CREATE MEET.TXT
John, Residents in the apartment complex will hold their annual meeting
this evening.  We hope to see you there, Regards, Elwood
Ctrl/Z
```

The CREATE command in this example creates a text file named MEET.TXT in your default directory. The text file MEET.TXT contains the lines that follow until the Ctrl/Z.

2.
```
$ CREATE A.DAT, B.DAT
Input line one for A.DAT...
Input line two for A.DAT...
   .
   .
   .
Ctrl/Z
Input line one for B.DAT...
```

```
Input line two for B.DAT...
   .
   .
   .
Ctrl/Z
$
```

After you enter the CREATE command from the terminal, the system reads input lines into the sequential file A.DAT until Ctrl/Z terminates the first input. The next set of input data is placed in the second file, B.DAT. Again, Ctrl/Z terminates the input.

3.
```
$ FILE = F$SEARCH("MEET.TXT")
$ IF FILE .EQS. ""
$ THEN CREATE MEET.TXT
  John, Residents in the apartment complex will hold their annual meeting
  this evening.  We hope to see you there, Regards, Elwood
$ ELSE TYPE MEET.TXT
$ ENDIF
$ EXIT
```

In this example, the command procedure searches the default disk and directory for the file MEET.TXT. If the command procedure determines that the file does not exist it creates a file named MEET.TXT using the CREATE command.

CREATE/DIRECTORY

Creates one or more new directories or subdirectories. The /DIRECTORY qualifier is required.

Requires write (W) access to the master file directory (MFD) to create a first-level directory. On a system volume, generally only users with a system user identification code (UIC) or the SYSPRV (system privilege) or BYPASS user privileges have write (W) access to the MFD to create a first-level directory.

Requires write (W) access to the lowest level directory that currently exists to create a subdirectory.

Format

CREATE/DIRECTORY directory-spec[,...]

Parameter

directory-spec[,...]
Specifies the name of one or more directories or subdirectories to be created. The directory specification optionally can be preceded by a device name (and colon [:]). The default is the current default directory. Wildcard characters are not allowed. When you create a subdirectory, separate the names of the directory levels with periods (.).

Note that it is possible to create a series of nested subdirectories with a single CREATE/DIRECTORY command. For example, [a.b.c] can be created, even though neither [a.b] nor [a] exists at the time the command is entered. Each subdirectory will be created, starting with the highest level and proceeding downward.

Description

The CREATE/DIRECTORY command creates new directories as well as subdirectories. Special privileges are needed to create new first-level directories. (See the restrictions noted above.) Generally, users have sufficient privileges to create subdirectories in their own directories. Use the SET DEFAULT command to move from one directory to another.

Qualifiers

/ALLOCATION=n
This qualifier applies only to Files–11 ODS–2 volumes. It specifies the initial number of blocks to be allocated to each of the specified directories. The default allocation is 1 block.

This qualifier is useful for creating large directories, for example MAIL.DIR;1. It can improve performance by avoiding the need for later dynamic expansion of the directory.

/LOG
/NOLOG (default)
Controls whether the CREATE/DIRECTORY command displays the directory specification of each directory after creating it.

/OWNER_UIC=option

Requires SYSPRV (system privilege) privilege for a user identification code (UIC) other than your own.

Specifies the owner UIC for the directory. The default is your UIC. You can specify the keyword PARENT in place of a UIC to mean the UIC of the parent (next-higher-level) directory. If a user with privileges creates a subdirectory, by default, the owner of the subdirectory will be the owner of the parent directory (or the owner of the MFD, if creating a main level directory). If you do not specifiy the /OWNER_UIC qualifier when creating a directory, the command assigns ownership as follows: (1) if you specify the directory name in either alphanumeric or subdirectory format, the default is your UIC (unless you are privileged in which case the UIC defaults to the parent directory); (2) if you specify the directory in UIC format, the default is the specified UIC.

Specify the UIC by using standard UIC format as described in the *OpenVMS User's Manual*.

/PROTECTION=(ownership[:access][,...])

Specifies protection for the directory.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), write (W), execute (E), or delete (D).

The default protection is the protection of the parent directory (the next-higher level directory, or the master directory for top-level directories) minus any delete (D) access.

If you are creating a first-level directory, then the next-higher-level directory is the MFD. (The protection of the MFD is established by the INITIALIZE command.)

For more information on specifying protection code, see the *OpenVMS Guide to System Security*.

/VERSION_LIMIT=n

Specifies the number of versions of any one file that can exist in the directory. If you exceed the limit, the system deletes the lowest numbered version. A specification of 0 means no limit. The maximum number of versions allowed is 32,767. The default is the limit for the parent (next-higher-level) directory.

When you change the version limit setting, the new limit applies only to files created after the setting was changed. New versions of files created before the change are subject to the previous version limit.

/VOLUME=n

Requests that the directory file be placed on the specified relative volume of a multivolume set. By default, the file is placed arbitrarily within the multivolume set.

Examples

1.
```
$ CREATE/DIRECTORY/VERSION_LIMIT=2 $DISK1:[ACCOUNTS.MEMOS]
```

   In this example, the CREATE/DIRECTORY command creates a subdirectory named MEMOS in the ACCOUNTS directory on $DISK1. No more than two versions of each file can exist in the directory.

2.
```
$CREATE/DIRECTORY/PROTECTION=(SYSTEM:RWCD,OWNER:RWCD,GROUP,WORLD) -
_$[MALCOLM.SUB.HLP]
```

   In this example, the CREATE/DIRECTORY command creates a subdirectory named [MALCOLM.SUB.HLP]. The protection on the subdirectory allows read (R), write (W), execute (E), and delete (D) access for the system and owner categories, but prohibits all access for the group or world categories.

3.
```
$ CREATE/DIRECTORY DISK2:[MALCOLM]
```

   In this example, the CREATE/DIRECTORY command creates a directory named [MALCOLM] on the device DISK2. Special privileges are required to create a first-level directory.

4.
```
$ CREATE/DIRECTORY [MALCOLM.SUB]
$ SET DEFAULT [MALCOLM.SUB]
```

   In this example, the CREATE/DIRECTORY command creates a subdirectory named [MALCOLM.SUB]. This directory file is placed in the directory named [MALCOLM]. The command SET DEFAULT [MALCOLM.SUB] changes the current default directory to this subdirectory. All files subsequently created are cataloged in [MALCOLM.SUB].

5.
```
$ CREATE/DIRECTORY [FRED.SUB1.SUB2.SUB3]
```

   In this example, the CREATE/DIRECTORY command creates a top-level directory ([FRED]) and three subdirectories ([FRED.SUB1], [FRED.SUB1.SUB2], and [FRED.SUB1.SUB2.SUB3]).

CREATE/FDL

Invokes the Create/FDL utility, which uses the specifications in a File Definition Language (FDL) file to create a new, empty data file. The /FDL qualifier is required. For a complete description of the Create/FDL utility, see the *OpenVMS Record Management Utilities Reference Manual*.

Format

CREATE/FDL=fdl-filespec [filespec]

CREATE/NAME_TABLE

Creates a new logical name table. The /NAME_TABLE qualifier is required.

Format

CREATE/NAME_TABLE table-name

Parameter

table-name
Specifies a string of 1 to 31 characters that identifies the logical name table you are creating. The string can include alphanumeric characters, the dollar sign ($), and the underscore (_). This name is entered as a logical name in either the process directory logical name table (LNM$PROCESS_DIRECTORY) or the system directory logical name table (LNM$SYSTEM_DIRECTORY).

Description

The CREATE/NAME_TABLE command creates a new logical name table. The name of the table is contained within the LNM$PROCESS_DIRECTORY directory table if the table is process-private, and within the LNM$SYSTEM_ DIRECTORY directory table if the table is shareable.

Every new table has a parent table, which determines whether the new table is process-private or shareable. To create a process-private table, use the /PARENT_ TABLE qualifier to specify the name of a process-private table (the process directory table). To create a shareable table, specify the parent as a shareable table.

If you do not explicitly provide a parent table, the CREATE/NAME_TABLE command creates a process-private table whose parent is LNM$PROCESS_ DIRECTORY; that is, the name of the table is entered in the process directory.

Every table has a size quota. The quota may either constrain the potential growth of the table or indicate that the table's size can be virtually unlimited. The description of the /QUOTA qualifier explains how to specify a quota.

To specify an access mode for the table you are creating, use the /USER_MODE, the /SUPERVISOR_MODE, or the /EXECUTIVE_MODE qualifier. If you specify more than one of these qualifiers, only the last one entered is accepted. If you do not specify an access mode, then a supervisor-mode table is created.

To delete a logical name table, use the DEASSIGN command, specify the name of the table you want to delete, and use the /TABLE qualifier to specify the directory table where the name of the table was entered.

For more information about logical name tables, see the *OpenVMS System Manager's Manual*.

Qualifiers

/ATTRIBUTES[=(keyword[,...])]
Specifies attributes for the logical name table. If you specify only one keyword, you can omit the parentheses. If you do not specify the /ATTRIBUTES qualifier, no attributes are set.

You can specify the following keywords for attributes:

CONFINE — Does not copy the table name or the logical names contained in the table into a spawned subprocess; used only when creating a private logical name table. If a table is created with the CONFINE attribute, all names subsequently entered into the table are also confined.

NO_ALIAS — No identical names (either logical names or names of logical name tables) may be created in an outer (less privileged) mode in the current directory. If you do not specify the NO_ALIAS attribute, then the table may be ''aliased'' by an identical name created in an outer access mode. Deletes any previously created identical table names in an outer access mode in the same logical name table directory.

SUPERSEDE — Creates a new table that supersedes any previous (existing) table that contains the name, access mode, and directory table that you specify. The new table is created regardless of whether the previous table exists. (If you do not specify the SUPERSEDE attribute, the new table is not created if the previous table exists.)

If you specify or accept the default for the qualifier /LOG, you receive a message indicating the result.

/EXECUTIVE_MODE

Requires SYSNAM (system logical name) privilege.

Creates an executive-mode logical name table. If you specify executive mode, but do not have SYSNAM privilege, a supervisor-mode logical name table is created.

/LOG (default)
/NOLOG

Controls whether an informational message is generated when the SUPERSEDE attribute is specified, or when the table already exists but the SUPERSEDE attribute is not specified. The default is the /LOG qualifier; that is, the informational message is displayed.

/PARENT_TABLE=table

Requires either create (C) access to the parent table and write (W) access to the system directory or the SYSNAM privilege.

Specifies the name of the parent table. The parent table determines whether a table is private or shareable; it also determines the size quota of the table. If you do not specify a parent table, the default table is LNM$PROCESS_DIRECTORY. A shareable table has LNM$SYSTEM_DIRECTORY as its parent table. The parent table must have the same access mode or a higher level access mode than the one you are creating.

/PROTECTION=(ownership[:access][,...])

Applies the specified protection to shareable name tables.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), write (W), create (C), or delete (D).

For more information on specifying protection codes, see the *OpenVMS Guide to System Security*.

The /PROTECTION qualifier affects only shareable logical name tables; it does not affect process-private logical name tables.

/QUOTA=number-of-bytes
Specifies the size limit of the logical name table. The size of each logical name entered in the new table is deducted from this size limit. The new table's quota is statically subtracted from the parent table's quota holder. The parent table's quota holder is the first logical name table encountered when working upward in the table hierarchy that has an explicit quota and is therefore its own quota holder. If the /QUOTA qualifier is not specified or the size limit is 0, the parent table's quota holder becomes the new table's quota holder and space is dynamically withdrawn from it whenever a logical name is entered in this new table. If the table has no quota holder and you specify /QUOTA=0, the table has unlimited quota.

/SUPERVISOR_MODE (default)
Creates a supervisor-mode logical name table. If you do not specify a mode, a supervisor-mode logical name table is created.

/USER_MODE
Creates a user-mode logical name table. If you do not explicitly specify a mode, a supervisor-mode logical name table is created.

Note

User-mode logical names are automatically deleted when invoking and exiting a command procedure.

Examples

1.
```
$ CREATE/NAME_TABLE TEST_TAB
$ SHOW LOGICAL TEST_TAB
%SHOW-S-NOTRAN, no translation for logical name TEST_TAB
$ SHOW LOGICAL/TABLE=LNM$PROCESS_DIRECTORY TEST_TAB
```

In this example, the CREATE/NAME_TABLE command creates a new table called TEST_TAB. By default, the name of the table is entered in the process directory. The first SHOW LOGICAL command does not find the name TEST_TAB because it does not, by default, search the process directory table. You must use the /TABLE qualifier to request that the process directory be searched.

2.
```
$ CREATE/NAME_TABLE/ATTRIBUTES=CONFINE EXTRA
$ DEFINE/TABLE=EXTRA MYDISK DISK4:
$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$FILE_DEV -
_$ EXTRA, LNM$PROCESS, LNM$JOB, LNM$GROUP, LNM$SYSTEM
$ TYPE MYDISK:[COHEN]EXAMPLE1.LIS
```

This example creates a new logical name table called EXTRA that is created with the CONFINE attribute. Therefore, the EXTRA table and the names it contains will not be copied to subprocesses.

Next, the logical name MYDISK is placed into the table EXTRA. To use the name MYDISK in file specifications, you must make sure that the table EXTRA is searched when RMS parses file specifications. To do this, you can define a process-private version of the logical name LNM$FILE_DEV to include the name EXTRA as one of its equivalence strings. (The system uses LNM$FILE_DEV to determine the tables to search during logical name translation for device or file specifications, and will use the process-private version of the logical name before using the default system version.) After you define LNM$FILE_DEV, the system searches the following tables during logical name translation: EXTRA, your process table, your job table, your group table, and the system table. Now, you can use the name MYDISK in a file specification and the equivalence string DISK4 will be substituted.

CREATE/TERMINAL

Creates a window that emulates another terminal type.

Note

At present, only DECterm windows are available with this command.

Format

CREATE/TERMINAL [command-string]

Parameter

command-string
Specifies a command string that is to be executed in the context of the created subprocess. You cannot specify this parameter with the /DETACH or the /NOPROCESS qualifier. The CREATE/TERMINAL command is used in much the same way as the SPAWN command.

Description

The CREATE/TERMINAL command creates a subprocess of your current process. When the subprocess is created, the process-permanent open files and any image or procedure context are *not* copied from the parent process. The subprocess is set to command level 0 (DCL level with the current prompt).

If you do not specify the /PROCESS qualifier, the name of this subprocess is composed of the same base name as the parent process and a unique number. For example, if the parent process name is SMITH, the subprocess name can be SMITH_1, SMITH_2, and so on.

The LOGIN.COM file of the parent process is not executed for the subprocess, because the context is copied separately, allowing quicker initialization of the subprocess. When the /WAIT qualifier is in effect, the parent process remains in hibernation until the subprocess terminates and returns control to the parent by using the ATTACH command.

You should use the LOGOUT command to terminate the subprocess and return to the parent process. You can also use the ATTACH command to transfer control of the terminal to another process in the subprocess tree, including the parent process. (The SHOW PROCESS/SUBPROCESS command displays the process in the subprocess tree and points to the current process.)

Note

Because a tree of subprocesses can be established using the CREATE /TERMINAL command, you must be careful when terminating any process in the tree. When a process is terminated, all the subprocesses below that point in the tree are automatically terminated. For example, the SPAWN/NOWAIT CREATE/TERMINAL command creates a subprocess that exits as soon as the DECterm window is created. Once this process exits, the DECterm window disappears. Instead, use the SPAWN/NOWAIT CREATE/TERMINAL/WAIT command to allow the process to continue.

Qualifiers with the CREATE/TERMINAL command must directly follow the command verb. The *command-string* parameter begins after the last qualifier and continues to the end of the command line.

Qualifiers

/APPLICATION_KEYPAD
Sets the APPLICATION_KEYPAD terminal characteristic in the created terminal window. If the /APPLICATION_KEYPAD or the /NUMERIC_KEYPAD qualifier is not specified, the default is to inherit the characteristic from the parent. (See also /NUMERIC_KEYPAD.)

/BIG_FONT
Specifies that the big font (as specified in resource files) be selected when the created terminal window is initialized. It is an error to specify the /BIG_FONT qualifier in combination with the /LITTLE_FONT qualifier. If you do not specify either the /BIG_FONT or the /LITTLE_FONT qualifier, the initial font is the big font.

/BROADCAST
/NOBROADCAST
Determines whether the terminal window is created with broadcast messages enabled. If neither qualifier is specified, the created terminal window inherits the broadcast characteristic of the parent.

/CARRIAGE_CONTROL
/NOCARRIAGE_CONTROL
Determines whether carriage-return and line-feed characters are prefixed to the subprocess's prompt string. By default, the CREATE/TERMINAL command copies the current setting of the parent process. The CARRIAGE_CONTROL qualifier is used only with the /NODETACH qualifier.

/CLI=cli-filespec
/NOCLI
Specifies the name of a command language interpreter (CLI) to be used by the subprocess. The default CLI is the same as that of the parent process (defined in SYSUAF). If you specify the /CLI qualifier, the attributes of the parent process are copied to the subprocess. The CLI you specify must be located in SYS$SYSTEM and have the file type EXE. This qualifier is used only with the /NODETACH qualifier.

/CONTROLLER=filespec
Specifies the name of the terminal window controller image. This name allows the CREATE/TERMINAL command to create a window on a variant controller, such as for a language not supported by the base product. For a DECterm window, the default is SYS$SYSTEM:DECW$TERMINAL.EXE. The device and directory default to SYS$SYSTEM and the file type defaults to EXE.

Note

The ''name'' field of the file name as returned by $PARSE is used to form the mailbox logical name. For example, if the file ''name'' is DECW$TERMINAL, the mailbox logical name will be DECW$TERMINAL_MAILBOX_node::0.0. For backward compatibility, the controller also defines a logical name DECW$DECTERM_MAILBOX_ host::0.0 to point to the same mailbox.

/DEFINE_LOGICAL=({logname, TABLE=tablename} [,...])
Specifies one or more logical names that are set to the name of the created pseudo terminal device. Each element in the list is either a logical name or TABLE= followed by the name of a logical name table in which all subsequent logical names will be entered. The default is the process logical name table.

/DETACH
/NODETACH (default)
Determines whether the created terminal process is detached or a subprocess of the current process. The /DETACH qualifier cannot be used with the *command-string* parameter.

/DISPLAY=display-name
Specifies the name of the display on which to create the terminal window. If this parameter is omitted, the DECW$DISPLAY logical name is used.

/ESCAPE
/NOESCAPE
Sets or clears the ESCAPE characteristic of the created terminal window. The default is to inherit the characteristic of the parent.

/FALLBACK
/NOFALLBACK
Sets or clears the FALLBACK characteristic of the created terminal window. The default is to inherit the characteristic of the parent.

/HOSTSYNC (default)
/NOHOSTSYNC
Sets or clears the HOSTSYNC characteristic of the created terminal window. The default is to inherit the characteristic of the parent.

/INPUT=filespec
Specifies an alternate input file or device to use as SYS$INPUT for the new process. The default is to use the created terminal window for input. This qualifier can be used with or without the /DETACH qualifier.

/INSERT
Creates the terminal window with insert mode as the default for line editing. If the /INSERT or the /OVERSTRIKE qualifier is not specified, the default is to inherit the characteristic from the parent. (See also /OVERSTRIKE.)

/KEYPAD (default)
/NOKEYPAD
Determines whether keypad definitions and the current keypad state are copied from the parent process. This qualifier is used only with the /NODETACH qualifier.

/LINE_EDITING
/NOLINE_EDITING
Determines whether the terminal window is created with line editing enabled. If neither qualifier is specified, the created terminal window inherits the line editing characteristic of the parent.

/LITTLE_FONT
Specifies that the little font (as specified in resource files) be selected when the created terminal window is initialized. It is an error to specify the /LITTLE_ FONT qualifier in combination with the /BIG_FONT qualifier. If you do not specify either the /BIG_FONT or the /LITTLE_FONT qualifier, the initial font is the big font.

/LOGGED_IN (default)
/NOLOGGED_IN
Determines whether a prompt for a user name and password are supplied (/NOLOGGED_IN) or the created terminal window is logged in automatically (/LOGGED_IN). This qualifier is used only with the /DETACH qualifier.

/LOGICAL_NAMES (default)
/NOLOGICAL_NAMES
Determines whether the created terminal window inherits the parent's logical names. This qualifier is used only with the /NODETACH qualifier.

/NOTIFY
/NONOTIFY (default)
Determines whether a notification message is broadcast to the parent when the created terminal window exits. This qualifier is used only with the /NODETACH qualifier.

/NUMERIC_KEYPAD
Sets the NUMERIC_KEYPAD terminal characteristic in the created terminal window. If the /NUMERIC_KEYPAD or the /APPLICATION_KEYPAD qualifier is not specified, the default is to inherit the characteristic from the parent. (See also /APPLICATION_KEYPAD.)

/OVERSTRIKE
Creates the terminal window with overstrike mode as the default for line editing. If the /OVERSTRIKE or the /INSERT qualifier is not specified, the default is to inherit the characteristic from the parent. (See also /INSERT.)

/PASTHRU
/NOPASTHRU
Sets or clears the PASTHRU characteristic in the created terminal window. The default is to inherit the characteristic of the parent.

/PROCESS (default)
/PROCESS=process-name
/NOPROCESS
Specifies the name of the process or subprocess to be created. The /NOPROCESS qualifier causes a window to be created without a process. You can log in from this window.

If you specify the /PROCESS qualifier without a process name, a unique process name is assigned with the same base name as the parent process and a unique number. The default process name format is *username_n*. If you specify a process name that already exists, an error message is displayed. This qualifier is used with either the /DETACH or the /NODETACH qualifier.

/PROMPT=prompt
Specifies the prompt string of the created terminal window. This qualifier is used only with the /NODETACH qualifier.

/READSYNC
/NOREADSYNC
Sets or clears the READSYNC terminal characteristic in the created terminal window. The default is to inherit the characteristic from the parent.

/RESOURCE_FILE=filespec
Specifies that the created terminal window use the resource file ''filespec'' instead of the default resource file, DECW$USER_DEFAULTS:DECW$TERMINAL_ DEFAULT.DAT.

/SYMBOLS (default)
/NOSYMBOLS
Determines whether the subprocess inherits the parent's DCL symbols. This qualifier is used only with the /NODETACH qualifier.

/TABLE=command-table
Specifies the name of an alternate command table to be used by the subprocess. This qualifier is used only with the /NODETACH qualifier.

/TTSYNC
/NOTTSYNC
Sets or clears the TTSYNC terminal characteristic in the created terminal window; the default is to inherit the characteristic of the parent.

/TYPE_AHEAD
/NOTYPE_AHEAD
Sets or clears the TYPE_AHEAD terminal characteristic in the created terminal window. The default is to inherit the characteristic of the parent.

/WAIT
/NOWAIT (default)
Requires that you wait for the subprocess to terminate before you enter another DCL command. The /NOWAIT qualifier allows you to enter new commands while the subprocess is running. This qualifier is used only with the /NODETACH qualifier.

/WINDOW_ATTRIBUTES=(parameter [,...])
Specifies initial attributes for the created terminal window to override the defaults read from the resource file. These parameters include:

Parameter	Description
BACKGROUND	The background color.
FOREGROUND	The foreground color.
WIDTH	The width, in pixels.
HEIGHT	The height, in pixels.
X_POSITION	The x-position, in pixels.
Y_POSITION	The y-position, in pixels.
ROWS	The number of rows in the window, in character cells. If the Auto Resize Window option is enabled, the ROWS and COLUMNS parameters override the size specified by the WIDTH and HEIGHT parameters.
COLUMNS	The number of columns in the window, in character cells. If the Auto Resize Window option is enabled, the ROWS and COLUMNS parameters override the size specified by the WIDTH and HEIGHT parameters.
INITIAL_STATE	The initial state of the window, either ICON or WINDOW.
TITLE	A character string specifying the window title.
ICON_NAME	A character string specifying the window icon name.
FONT	The name of the font to be used in the window. If you specify the /LITTLE_FONT qualifier, or omit both the /LITTLE_ FONT and /BIG_FONT qualifiers, this overrides the name of the little font that is set in the resource files; otherwise it overrides the name of the big font. The font name can be a logical name, and it can be (but does not have to be) the base font in a complete font set.

Example

```
$ CREATE/TERMINAL=DECTERM/DETACH -
_$ /DISPLAY=MYNODE::0 -
_$ /WINDOW_ATTRIBUTES=( -
_$ ROWS=36, -
_$ COLUMNS=80, -
_$ TITLE="REMOTE TERMINAL", -
_$ ICON_NAME="REMOTE TERMINAL" )
```

In this example, the command creates a detached process in a DECterm window on node MYNODE:: that is 36 rows by 80 columns and has its title and icon name set to ‘‘Remote terminal’’.

```
$ CREATE/TERMINAL=DECTERM -
$_ /NOPROCESS -
$_ /DEFINE_LOGICAL=(TABLE=LNM$GROUP,DBG$INPUT,DBG$OUTPUT)
```

In this example, the command creates a DECterm with no associated process. The command defines DBG$INPUT and DBG$OUTPUT in the group table as the new terminal for the purposes of debugging a problem with a detached process that is subsequently created.

DEALLOCATE

Makes an allocated device available to other processes (but does not deassign any logical name associated with the device). DEALLOCATE does not deallocate devices that are in use.

Format

DEALLOCATE device-name[:]

Parameter

device-name[:]
Name of the device to be deallocated. The device name can be a physical device name or a logical name that is not in use. On a physical device name, the controller defaults to A and the unit to 0. This parameter is incompatible with the /ALL qualifier.

Qualifier

/ALL
Deallocates all devices currently allocated by your process that are not in use. This qualifier is incompatible with the *device-name* parameter.

Examples

1.
```
$ DEALLOCATE  DMB1:
```

   In this example, the DEALLOCATE command deallocates unit 1 of the RK06/RK07 devices on controller B.

2.
```
$ ALLOCATE   MT:   TAPE
%DCL-I-ALLOC, _MTB1:  allocated
   .
   .
   .
$ DEALLOCATE  TAPE:
```

   In this example, the ALLOCATE command requests that any magnetic tape drive be allocated and assigns the logical name TAPE to the device. The response to the ALLOCATE command indicates the successful allocation of the device MTB1. The DEALLOCATE command specifies the logical name TAPE to release the tape drive.

3.
```
$ DEALLOCATE/ALL
```

   In this example, the DEALLOCATE command deallocates all devices that are currently allocated.

DEASSIGN

Cancels a logical name assignment that was made with one of the following commands: ALLOCATE, ASSIGN, DEFINE, or MOUNT. The DEASSIGN command also deletes a logical name table that was created with the CREATE /NAME_TABLE command.

Format

DEASSIGN [logical-name[:]]

Parameter

logical-name[:]
Specifies the logical name to be deassigned. Logical names can have from 1 to 255 characters. If the logical name contains any characters other than alphanumerics, dollar signs ($), or underscores (_), enclose it in quotation marks (‘‘ ’’). The *logical-name* parameter is required unless you use the /ALL qualifier.

If the *logical-name* parameter ends with a colon (:), the command interpreter ignores the colon. (Note that the ASSIGN and ALLOCATE commands remove a trailing colon, if present, from a logical name before placing the name in a logical name table.) If a colon is present in the logical name, you must type two colons in the *logical-name* parameter of the DEASSIGN command (for example, DEASSIGN FILE::).

To delete a logical name table, specify the table name as the *logical-name* parameter. You must also use the /TABLE qualifier to indicate the logical name directory table where the table name is entered.

Description

The DEASSIGN command cancels a logical name assignment that was made with one of the following commands: ALLOCATE, ASSIGN, DEFINE, or MOUNT. The DEASSIGN command also deletes a logical name table that was created with the CREATE/NAME_TABLE command. You can use the /ALL qualifier with DEASSIGN to cancel all logical names in a specified table. If you use the /ALL qualifier and do not specify a table, then all names in the process table (except names created by the command interpreter) are deassigned; that is, all names entered at the indicated access mode or an outer access mode are deassigned.

To specify the logical name table from which you want to deassign a logical name, use the /PROCESS, the /JOB, the /GROUP, the /SYSTEM, or the /TABLE qualifier. If you enter more than one of these qualifiers, only the last one entered is accepted. If entries exist for the specified logical name in more than one logical name table, the name is deleted from only the last logical name table specified on the command line. If you do not specify a logical name table, the default is the /TABLE=LNM$PROCESS qualifier.

To delete a shareable logical name, you need write (W) access to the logical name table. To delete a shareable logical name table, you need write (W) access to the parent table and delete (D) access to the target logical name table.

To specify the access mode of the logical name you want to deassign, use the /USER_MODE, the /SUPERVISOR_MODE, or the /EXECUTIVE_MODE qualifier. If you enter more than one of these qualifiers, only the last one is accepted. If you do not specify a mode, the DEASSIGN command deletes a supervisor-mode name. When you deassign a logical name, any identical names created with outer access modes in the same logical name table are also deleted.

You must have SYSNAM (system logical name) privilege to deassign an executive-mode logical name.

If you specify the /EXECUTIVE_MODE qualifier and you do not have SYSNAM privilege, then the DEASSIGN command ignores the qualifier and attempts to deassign a supervisor-mode logical name.

All process-private logical names and logical name tables are deleted when you log out of the system. User-mode entries within the process logical name table are deassigned when any image exits. The logical names in the job table, and the job table itself, are deleted when you log off the system.

Names in all other shareable logical name tables remain there until they are explicitly deassigned, regardless of whether they are user-, supervisor-, or executive-mode names. You must have write (W) access to a shareable logical name table to delete any name in that table.

If you delete a logical name table, all the logical names in the table are also deleted. Also, any descendant tables are deleted. To delete a shareable logical name table, you must have delete (D) access to the table.

Qualifiers

/ALL

Deletes all logical names in the same or an outer (less privileged) access mode. If no logical name table is specified, the default is the process table, LNM$PROCESS. If you specify the /ALL qualifier, you cannot enter a *logical-name* parameter.

/EXECUTIVE_MODE

Requires SYSNAM (system logical name) privilege to deassign executive-mode logical names.

Deletes only entries that were created in the specified mode or an outer (less privileged) mode. If you do not have SYSPRV (system privilege) privilege for executive mode, a supervisor-mode operation is assumed.

/GROUP

Requires GRPNAM (group logical name) or SYSPRV privilege to delete entries from the group logical name table.

Indicates that the specified logical name is in the group logical name table. The /GROUP qualifier is synonymous with the /TABLE=LNM$GROUP qualifier.

/JOB

Indicates that the specified logical name is in the jobwide logical name table. The /JOB qualifier is synonymous with the /TABLE=LNM$JOB qualifier. If you do not explicitly specify a logical name table, the default is the /PROCESS qualifier.

You should not deassign jobwide logical name entries that were made by the system at login time, for example, SYS$LOGIN, SYS$LOGIN_DEVICE, and SYS$SCRATCH. However, if you assign new equivalence names for these logical names (that is, create new logical names in outer access modes), you can deassign the names you explicitly created.

/PROCESS (default)
Indicates that the specified logical name is in the process logical name table. The /PROCESS qualifier is synonymous with the /TABLE=LNM$PROCESS qualifier.

You cannot deassign logical name table entries that were made by the command interpreter, for example, SYS$INPUT, SYS$OUTPUT, and SYS$ERROR. However, if you assign new equivalence names for these logical names (that is, you create new logical names in outer access modes), you can deassign the names you explicitly created.

/SUPERVISOR_MODE (default)
Deletes entries in the specified logical name table that were created in supervisor mode. If you specify the /SUPERVISOR_MODE qualifier, the DEASSIGN command also deassigns user-mode entries with the same name.

/SYSTEM
Indicates that the specified logical name is in the system logical name table. The /SYSTEM qualifier is synonymous with the /TABLE=LNM$SYSTEM qualifier.

/TABLE=name
Specifies the table from which the logical name is to be deleted. Defaults to LNM$PROCESS. The table can be the process, group, job, or system table, one of the directory tables, or the name of a user-created table. (The process, job, group, and system logical name tables should be referred to by the logical names LNM$PROCESS, LNM$JOB, LNM$GROUP, and LNM$SYSTEM, respectively.)

The /TABLE qualifier also can be used to delete a logical name table. To delete a process-private table, enter the following command:

```
$ DEASSIGN/TABLE=LNM$PROCESS_DIRECTORY table-name
```

To delete a shareable table, enter the following command:

```
$ DEASSIGN/TABLE=LNM$SYSTEM_DIRECTORY table-name
```

To delete a shareable logical name table, you must have delete (D) access to the table or write (W) access to the directory table in which the name of the shareable table is cataloged.

If you do not explicitly specify the /TABLE qualifier, the default is the /TABLE=LNM$PROCESS qualifier.

/USER_MODE
Deletes entries in the process logical name table that were created in user mode. If you specify the /USER_MODE qualifier, the DEASSIGN command can deassign only user-mode entries. Also, user-mode logical names are automatically deleted when invoking and exiting a command procedure.

Examples

1.
```
$ DEASSIGN MEMO
```

   The DEASSIGN command in this example deassigns the process logical name MEMO.

2.
```
$ DEASSIGN/ALL
```

   The DEASSIGN command in this example deassigns all process logical names that were created in user and supervisor mode. This command does not, however, delete the names that were placed in the process logical name table in executive mode by the command interpreter (for example, SYS$INPUT, SYS$OUTPUT, SYS$ERROR, SYS$DISK, and SYS$COMMAND).

3.
```
$ DEASSIGN/TABLE=LNM$PROCESS_DIRECTORY TAX
```

   The DEASSIGN command in this example deletes the logical name table TAX, and any descendant tables. When you delete a logical name table, you must specify either the /TABLE=LNM$PROCESS_DIRECTORY or the /TABLE=LNM$SYSTEM_DIRECTORY qualifier, because the names of all tables are contained in these directories.

4.
```
$ ASSIGN USER_DISK:   COPY
$ DEASSIGN COPY
```

   The ASSIGN command in this example equates the logical name COPY with the device USER_DISK and places the names in the process logical name table. The DEASSIGN command deletes the logical name.

5.
```
$ DEFINE SWITCH:  TEMP
$ DEASSIGN SWITCH::
```

   The DEFINE command in this example places the logical name SWITCH: in the process logical name table. The trailing colon is retained as part of the logical name. Two colons are required on the DEASSIGN command to delete this logical name because the DEASSIGN command removes one trailing colon, and the other colon is needed to match the characters in the logical name.

6.
```
$ ASSIGN/TABLE=LNM$GROUP DBA1: GROUP_DISK
$ DEASSIGN/PROCESS/GROUP GROUP_DISK
```

   The ASSIGN command in this example places the logical name GROUP_ DISK in the group logical name table. The DEASSIGN command specifies conflicting qualifiers; because the /GROUP qualifier is last, the name is successfully deassigned.

7.
```
$ ASSIGN DALLAS::USER_DISK:   DATA
    .
    .
    .
$ DEASSIGN DATA
```

   The ASSIGN command in this example associates the logical name DATA with the device specification USER_DISK on remote node DALLAS. Subsequent references to the logical name DATA result in references to the disk on the remote node. The DEASSIGN command cancels the logical name assignment.

DEASSIGN/QUEUE

Deassigns a logical queue from a printer or terminal queue and stops the logical queue. Cannot be used with batch queues.

Requires manage (M) access to the queue.

Format

DEASSIGN/QUEUE logical-queue-name[:]

Parameter

logical-queue-name[:]
Specifies the name of the logical queue that you want to deassign from a specific printer or terminal queue.

Description

Once you enter the DEASSIGN/QUEUE command, the jobs in the logical queue remain pending until the queue is reassigned to another printer queue or device with the ASSIGN/QUEUE command.

Example

```
$ ASSIGN/QUEUE LPA0  ASTER
   .
   .
   .
$ DEASSIGN/QUEUE  ASTER
$ ASSIGN/MERGE LPB0  ASTER
```

The ASSIGN/QUEUE command in this example associates the logical queue ASTER with the print queue LPA0. Later, you deassign the logical queue with the DEASSIGN/QUEUE command. The ASSIGN/MERGE command reassigns the jobs from ASTER to the print queue LPB0.

DEBUG

Invokes the OpenVMS Debugger.

For a complete description of the OpenVMS Debugger, see the *OpenVMS Debugger Manual*.

To get help on debugger commands from DCL level, type the following command:

```
$ HELP/LIBRARY=SYS$HELP:DBG$HELP DEBUG
```

Format

DEBUG

Heap Analyzer

The Heap Analyzer provides a graphical representation of memory use in real time. This allows you to quickly identify inefficient memory usage in your application such as allocations that are made too often, memory blocks that are too large, fragmentation, or memory leaks.

For details on running the Heap Analyzer from within the debugger, see the *OpenVMS Debugger Manual*.

To run the standalone Heap Analyzer, issue the following commands:

```
$ DEFINE/USER LIBRTL SYS$LIBRARY:LIBRTL_INSTRUMENTED
$ RUN/NODEBUG program
```

Qualifiers

/KEEP

Invokes the Kept Debugger. The Kept Debugger includes a Run/Rerun capability that allows you to debug an image multiple times or debug a series of distinct images without exiting the debugger.

Issuing the DEBUG/KEEP command is the only way to invoke the Kept Debugger.

/RESUME (default)

Reinvokes the non-Kept Debugger after a Ctrl/Y key sequence has interrupted the execution of a program you are debugging. (The interrupted program must not have been linked with a /NOTRACEBACK qualifier on the LINK command.)

If you issue the DEBUG/RESUME command without a previous Ctrl/Y key sequence, no action occurs.

Examples

1.
```
$ FORTRAN/DEBUG/NOOPTIMIZE WIDGET
$ LINK/DEBUG WIDGET
$ RUN WIDGET

                [ Debugger Banner and Version ]

%DEBUG-I-INITIAL, language is FORTRAN, module set to WIDGET
DBG>
```

   The FORTRAN and LINK commands both specify the /DEBUG qualifier to compile the program WIDGET.FOR with debugger symbol table information.

Because the program has been compiled and linked with debug information, the debugger is automatically invoked by the image activator upon starting the program with the RUN command. No program code has yet been executed when the debugger is invoked.

2.
```
$ FORTRAN/DEBUG/NOOPTIMIZE WIDGET
$ LINK/DEBUG WIDGET
$ RUN/NODEBUG WIDGET

  NAME:
  NAME:
  NAME:
^Y
$ DEBUG/RESUME

              [ Debugger Banner and Version ]

  %DEBUG-I-INITIAL, language is FORTRAN, module set to WIDGET
  DBG>
```

The FORTRAN and LINK commands both specify the /DEBUG qualifier to compile the program WIDGET.FOR with debugger symbol table information. The RUN command begins execution of the image WIDGET.EXE, which loops uncontrollably. Ctrl/Y interrupts the program, and the DEBUG/RESUME command gives control to the debugger.

3.
```
$ CC/DEBUG/NOOPTIMIZE ECHOARGS
$ LINK/DEBUG ECHOARGS
$ ECHO == "$ sys$disk:[]echoargs.exe"
$ DEBUG/KEEP

                    [ Debugger Banner and Version ]

DBG> RUN/COMMAND="ECHO"/ARGUMENTS="fa sol la mi"
%DEBUG-I-INITIAL, language is C, module set to ECHOARGS
%DEBUG-I-NOTATMAIN, type GO to get to start of main program
DBG>
   .
   .
   .
DBG> RERUN/ARGUMENTS="fee fii foo fum"
%DEBUG-I-INITIAL, language is C, module set to ECHOARGS
%DEBUG-I-NOTATMAIN, type GO to get to start of main program
DBG>
   .
   .
   .
DBG> RUN/ARGUMENTS="a b c" ECHOARGS
%DEBUG-I-INITIAL, language is C, module set to ECHOARGS
%DEBUG-I-NOTATMAIN, type GO to get to start of main program
DBG>
```

The CC and LINK commands both specify the /DEBUG qualifier to compile the program ECHOARGS.C with debugger symbol table information.

The symbol definition command defines a foreign command for use during the debugging session.

The DEBUG/KEEP command invokes the Kept Debugger.

The first RUN command uses the /COMMAND qualifier to specify a foreign command to invoke the image file and the /ARGUMENTS qualifier to specify a string of arguments.

The RERUN command reinvokes the same image file and uses the /ARGUMENTS qualifier to specify a new string of arguments.

The second RUN command specifies a new image file and a new string of arguments.

4.
```
$ PASCAL/DEBUG/NOOPTIMIZE 8QUEENS
$ LINK/DEBUG 8QUEENS
$ DEFINE/USER LIBRTL SYS$LIBRARY:LIBRTL_INSTRUMENTED
$ RUN/NODEBUG 8QUEENS

        [Heap Analyzer window is displayed]
```

The PASCAL and LINK commands both specify the /DEBUG qualifier to compile the program 8QUEENS.PAS with debugger symbol table information.

The DEFINE command causes the Heap Analyzer to access a version of LIBRTL designed to collect memory allocation and deallocation information.

The RUN/NODEBUG command invokes the Heap Analyzer but not the Debugger.

DECK

Marks the beginning of an input stream for a command or program.

Format

DECK

Description

The DECK command marks the data that follows it as input for a command or program. The DECK command can be used only after a request to execute a command or program that requires input data.

In command procedures, this command is required when the first nonblank character in any data record in the input stream is a dollar sign. Also in command procedures, the DECK command must be preceded by a dollar sign; the dollar sign must be in the first character position (column 1) of the input record.

The DECK command defines an end-of-file (EOF) indicator only for a single data stream. Using the DECK command enables you to place data records beginning with dollar signs in the input stream. You can place one or more sets of data in the input stream following a DECK command, if each is terminated by an EOF indicator.

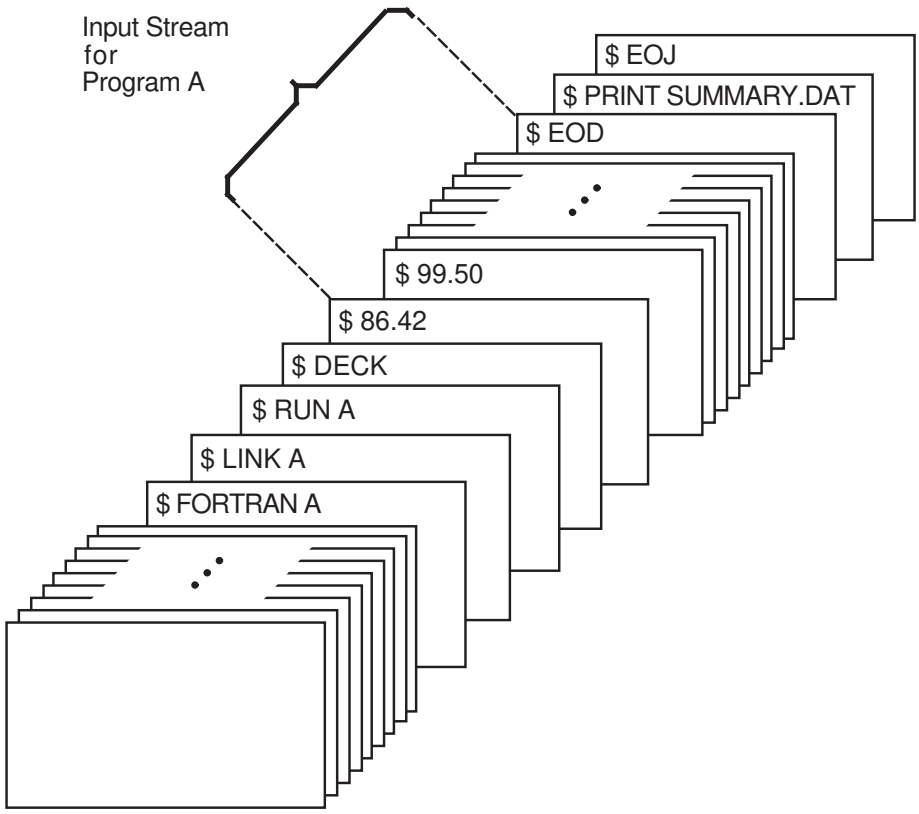
After an EOF indicator specified with the /DOLLARS qualifier is encountered, the EOF indicator is reset to the default, that is, to any record beginning with a dollar sign. The default is also reset if an actual EOF indicator occurs for the current command level.

Qualifier

/DOLLARS[=string]
Sets the EOF indicator to the specified string of 1 to 15 characters. Specify a string if the input data contains one or more records beginning with the string $EOD. Enclose the string in quotation marks (‘‘ ’’) if it contains literal lowercase letters, multiple blanks, or tabs. If you do not specify /DOLLARS or if you specify /DOLLARS without specifying a string, you must use the EOD command to signal the end-of-file (EOF).

Examples

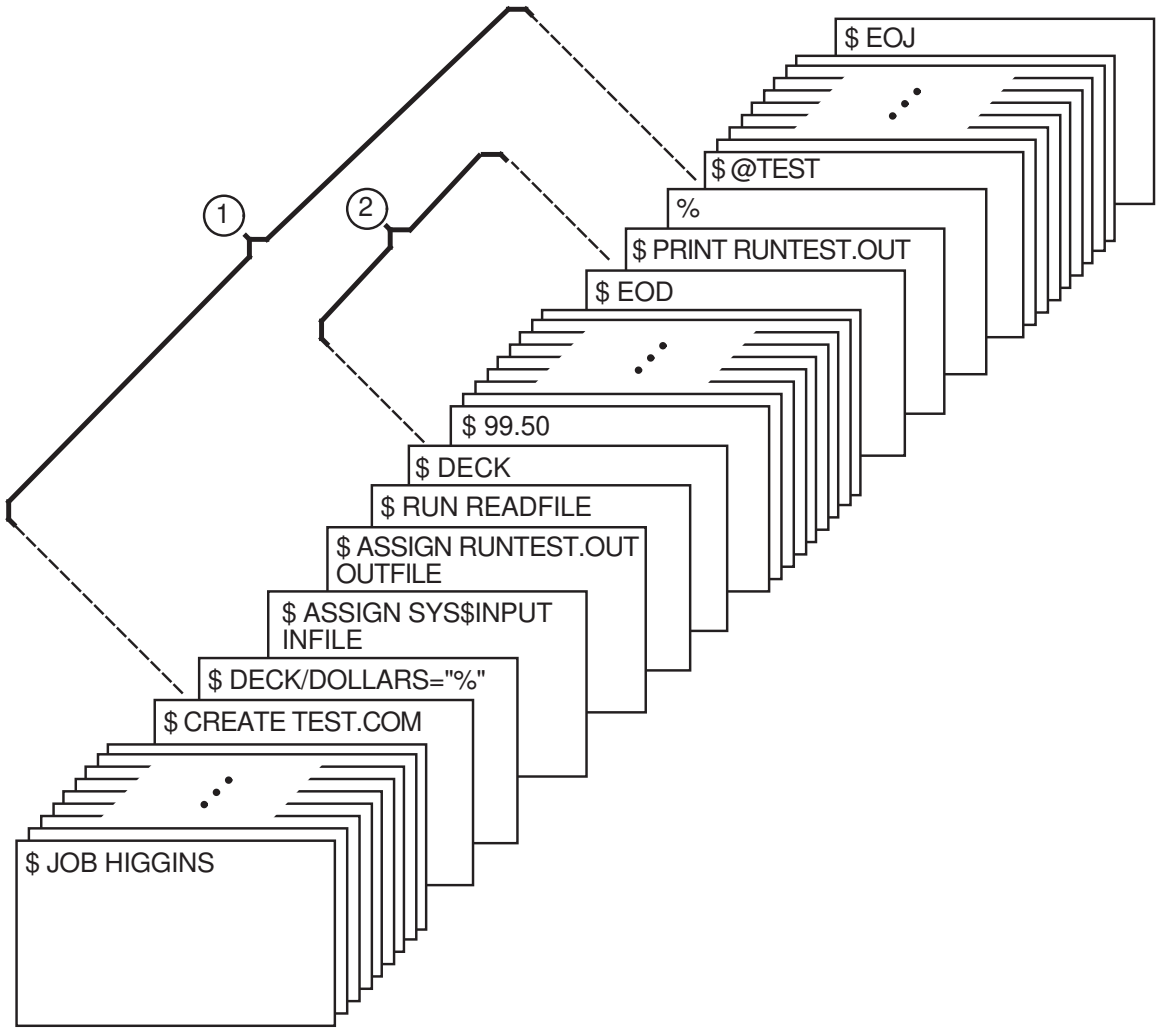
1.



ZK–0783–GE

In this example, the FORTRAN and LINK commands compile and link program A. When the program is run, any data the program reads from the logical device SYS$INPUT is read from the command stream. The DECK command indicates that the input stream can contain dollar signs in column 1 of the record. The EOD command signals end-of-file (EOF) for the data.

2.



(1) Input stream for CREATE command.

(2) Input stream for program READFILE.

ZK–0784–GE

The CREATE command in this example creates the command procedure file TEST.COM from lines entered into the input stream. The DECK/DOLLARS command indicates that the percent sign (%) is the EOF indicator for the CREATE command. This allows the string $EOD to be read as an input record, signaling the end of the input for the RUN command.

DEFINE

Associates an equivalence name with a logical name.

Format

DEFINE logical-name equivalence-name[,...]

Parameters

logical-name

Specifies the logical name string, which is a character string containing from 1 to 255 characters. The following rules apply:

- If you specify an existing logical name, the new equivalence names replace the existing equivalence name.
- If the logical name is to be entered into the process or system directory logical name tables (LNM$PROCESS_DIRECTORY, LNM$SYSTEM_DIRECTORY), then the name can only have from 1 to 31 alphanumeric characters, including the dollar sign ($) and underscore (_).
- If you specify a colon (:) at the end of a logical name, the DEFINE command saves the colon as part of the logical name. (This is in contrast to the ASSIGN command, which removes the colon before placing the name in a logical name table.) By default, the logical name is placed in the process logical name table.
- If the string contains any characters other than uppercase alphanumerics, the dollar sign, or the underscore character, enclose the string in quotation marks (‘‘ ’’). Use two sets of quotation marks (‘‘‘‘ ’’’’) to denote actual quotation marks. When you enclose a name in quotation marks, the case of alphabetic characters is preserved.

equivalence-name[,...]

Specifies a character string containing from 1 to 255 characters. The following rules apply:

- If the string contains any characters other than uppercase alphanumerics, the dollar sign, or the underscore character, enclose the string in quotation marks. Use two sets of quotation marks to denote an actual quotation mark. Specifying more than one equivalence name for a logical name creates a search list.
- When you specify an equivalence name that will be used as a file specification, you must include the punctuation marks (colons, brackets, periods) that would be required if the equivalence name were used directly as a file specification. Therefore, if you specify a device name as an equivalence name, you must terminate the equivalence name with a colon.

The DEFINE command allows you to assign the same logical name to more than one equivalence name. For example, you can use the same logical name to access different directories on different disks or to access different files in different directories.

When you specify more than one equivalence name for a logical name, you create a search list. See the *OpenVMS User's Manual* for more information on search lists.

Description

The DEFINE command creates an entry in a logical name table by defining a logical name to stand for one or more equivalence names. An equivalence name can be a device name, another logical name, a file specification, or any other string.

To specify the logical name table where you want to enter a logical name, use the /PROCESS, the /GROUP, the /SYSTEM, the /JOB, or the /TABLE qualifier. If you enter more than one of these qualifiers, only the last one entered is accepted. If you do not specify a table, the default is the /TABLE=LNM$PROCESS qualifier.

To specify the access mode of the logical name you are creating, use the /USER_MODE, the /SUPERVISOR_MODE, or the /EXECUTIVE_MODE qualifier. If you enter more than one of these qualifiers, only the last one entered is accepted. If you do not specify an access mode, a supervisor-mode name is created. You can create a logical name in the same mode as the table in which you are placing the name, or in an outer mode. (User mode is the outermost mode; executive mode is the innermost mode.)

You can enter more than one logical name with the same name in the same table, as long as each name has a different access mode. (However, if an existing logical name within a table has the NO_ALIAS attribute, you cannot use the same name to create a logical name in an outer mode in this table.)

If you create a logical name with the same name, in the same table, and in the same mode as an existing name, the new logical name assignment replaces the existing assignment.

You can also use the ASSIGN command to create logical names. To delete a logical name from a table, use the DEASSIGN command.

Note

Avoid assigning a logical name that matches the file name of an executable image in SYS$SYSTEM:. Such an assignment prohibits you from invoking that image.

For additional information on how to create and use logical names, see the *OpenVMS User's Manual*.

Qualifiers

/EXECUTIVE_MODE

Requires SYSNAM (system logical name) privilege to create an executive-mode logical name.

Creates an executive-mode logical name in the specified table.

If you specify the /EXECUTIVE_MODE qualifier and you do not have SYSNAM privilege, the DEFINE command ignores the qualifier and creates a supervisor-mode logical name. The mode of the logical name must be the same or less privileged than the mode of the table in which you are placing the name.

/GROUP

Requires GRPNAM (group logical name) or SYSNAM (system logical name) privilege to place a name in the group logical name table.

Places the logical name in the group logical name table. Other users who have the same group number in their user identification codes (UICs) can access the logical name. The /GROUP qualifier is synonymous with the /TABLE=LNM$GROUP qualifier.

/JOB
Places the logical name in the jobwide logical name table. All processes in the same job tree as the process that created the logical name can access the logical name. The /JOB qualifier is synonymous with the /TABLE=LNM$JOB qualifier.

/LOG (default)
/NOLOG
Displays a message when a new logical name supersedes an existing name.

/NAME_ATTRIBUTES[=(keyword[,...])]
Specifies attributes for a logical name. By default, no attributes are set. Possible keywords are as follows:

CONFINE	The logical name is not copied into a spawned subprocess. This qualifier is relevant only for logical names in a private table. The logical name inherits the CONFINE attribute from the logical name table where it is entered; if the logical name table is ''confined,'' then all names in the table are ''confined.''
NO_ALIAS	A logical name cannot be duplicated in the specified table in a less privileged access mode; any previously created identical names in an outer (less privileged) access mode within the specified table are deleted.

If you specify only one keyword, you can omit the parentheses. Only the attributes you specify are set.

/PROCESS (default)
Places the logical name in the process logical name table. The /PROCESS qualifier is synonymous with the /TABLE=LNM$PROCESS qualifier.

/SUPERVISOR_MODE (default)
Creates a supervisor-mode logical name in the specified table. The mode of the logical name must be the same as or less privileged than the mode of the table in which you are placing the name.

/SYSTEM
Requires write (W) access or SYSNAM (system logical name) privilege to place a name in the system logical name table.

Places the logical name in the system logical name table. All system users can access the logical name. The /SYSTEM qualifier is synonymous with the /TABLE=LNM$SYSTEM qualifier.

/TABLE=name
Requires write (W) access to the table to specify the name of a shareable logical name table.

Specifies the name of the logical name table in which the logical name is to be entered. You can use the /TABLE qualifier to specify a user-defined logical name table (created with the CREATE/NAME_TABLE command); to specify the process, job, group, or system logical name tables; or to specify the process or system logical name directory tables.

If you specify the table name using a logical name that has more than one translation, the logical name is placed in the first table found. For example, if you specify DEFINE/TABLE=LNM$FILE_DEV and LNM$FILE_DEV is equated to LNM$PROCESS, LNM$JOB, LNM$GROUP, and LNM$SYSTEM, then the logical name is placed in LNM$PROCESS.

The default is the /TABLE=LNM$PROCESS qualifier.

/TRANSLATION_ATTRIBUTES[=(keyword[,...])]
Equivalence-name qualifier.

Specifies one or more attributes that modify an equivalence string of the logical name. Possible keywords are as follows:

CONCEALED	Indicates that the equivalence string is the name of a concealed device. When a concealed device name is defined, the system displays the logical name, rather than the equivalence string, in messages that refer to the device.
TERMINAL	Logical name translation should terminate with the current equivalence string; indicates that the equivalence string should not be translated iteratively.

If you specify only one keyword, you can omit the parentheses. Only the attributes you specify are set.

Note that different equivalence strings of a logical name can have different translation attributes.

/USER_MODE
Creates a user-mode logical name in the specified table.

User-mode logical names created within the process logical name tables are used for the execution of a single image; for example, you can create a user-mode logical name to allow an image executing in a command procedure to redefine SYS$INPUT. User-mode entries are deleted from the process logical name table when any image executing in the process exits (that is, after a DCL command or user program that executes an image completes execution). Also, user-mode logical names are automatically deleted when invoking and exiting a command procedure.

Examples

1.
```
$ DEFINE/USER_MODE TM1 $DISK1:[ACCOUNTS.MEMOS]WATER.TXT
```

   In this example, the DEFINE command defines TM1 as equivalent to a file specification. After the next image runs, the logical name TM1 is automatically deassigned.

2.
```
$ DEFINE MEMO $DISK1:[ACCOUNTS.MEMO]
```

   In this example, the DEFINE command defines the logical name MEMO as equivalent to the partial file specification $DISK1:[ACCOUNTS.MEMO].

3.
```
$ DEFINE PROCESS_NAME  LIBRA
$ RUN WAKE
```

In this example, the DEFINE command places the logical name PROCESS_ NAME in the process logical name table with an equivalence name of LIBRA. The logical name is created in supervisor mode. The program WAKE translates the logical name PROCESS_NAME to perform some special action on the process named LIBRA.

4.
```
$ DEFINE TEMP:  XXX1:
    .
    .
    .
$ DEASSIGN TEMP::
```

In this example, the DEFINE command creates an equivalence name for the logical name TEMP: and places the name in the process logical name table. The colon is retained as part of the logical name. The DEASSIGN command deletes the logical name. Note that two colons are required on the logical name in the DEASSIGN command. One colon is deleted by the DEASSIGN command. The other colon is kept as part of the logical name.

5.
```
$ DEFINE PORTLAND PRTLND::YYY0:[DECNET.DEMO.COM]
```

In this example, the DEFINE command places the logical name PORTLAND in the process logical name table with an equivalence name of PRTLND::YYY0:[DECNET.DEMO.COM]. Subsequent references to the logical name PORTLAND result in the correspondence between the logical name PORTLAND and the node, disk, and subdirectory specified.

6.
```
$ DEFINE LOCAL "BOSTON""JOHN_SMITH JKS""::"
```

In this example, the DEFINE command places the logical name LOCAL in the process logical name table with a remote node equivalence name of BOSTON"JOHN_SMITH JKS"::. To satisfy conventions for local DCL command string processing, you must use three sets of quotation marks. The quotation marks ensure that access control information is enclosed in one set of quotation marks in the equivalence name.

7.
```
$ DEFINE MYDISK XXX0:[MYDIR], YYY0:[TESTDIR]
```

In this example, the DEFINE command places the logical name MYDISK in the process logical name table with two equivalence names: XXX0:[MYDIR] and YYY0:[TESTDIR].

8.
```
$  CREATE/NAME_TABLE TABLE1
$  DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$FILE_DEV -
_$ TABLE1,LNM$PROCESS,LNM$JOB,LNM$GROUP,LNM$SYSTEM
$  DEFINE/TABLE=TABLE1 -
_$ /TRANSLATION_ATTRIBUTES=CONCEALED   WORK_DISK DBA1:
```

In this example, the CREATE/NAME_TABLE command creates the process private logical name table TABLE1.

The first DEFINE command ensures that TABLE1 is searched first in any logical name translation of a device or file specification (because TABLE1 is the first item in the equivalence string for the logical name LNM$FILE_DEV, which determines the default search sequence of logical name tables whenever a device or file specification is translated).

The second DEFINE command assigns the logical name WORK_DISK to the physical device DBA1 and places the name in TABLE1. The logical name has the concealed attribute. Therefore, the logical name WORK_DISK is displayed in system messages.

9.
```
$ CREATE/NAME_TABLE SPECIAL
$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$FILE_DEV -
_$ SPECIAL,LNM$PROCESS,LNM$JOB,LNM$GROUP,LNM$SYSTEM
$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY TAB SPECIAL
$ DEFINE/TABLE=TAB REPORT [CHELSEA]STORES
$ SHOW LOGICAL/TABLE=SPECIAL REPORT
  "REPORT" = "[CHELSEA]STORES" (SPECIAL)
```

In this example, the CREATE/NAME_TABLE command is used to create a new logical name table called SPECIAL. This table is defined in the process directory, LNM$PROCESS_DIRECTORY.

The first DEFINE command ensures that SPECIAL is searched first in any logical name translation of a device or file specification (because SPECIAL is the first item in the equivalence string for the logical name LNM$FILE_DEV, which determines the default search sequence of logical name tables whenever a device or file specification is translated). The logical name LNM$FILE_DEV is placed in the process directory, LNM$PROCESS_DIRECTORY.

With the next DEFINE command, a new logical name, TAB, is defined. TAB translates to the string SPECIAL, which identifies a logical name table. You must define TAB in the process directory because it translates iteratively to a logical name table.

Next, the logical name REPORT is placed into the logical name table TAB. Because TAB translates to the table SPECIAL, the name REPORT is entered into SPECIAL table. The SHOW LOGICAL command verifies that the name REPORT has been entered into the table SPECIAL.

Note that you can redefine TAB so it translates to a different table. Therefore, if you run different programs that use the name TAB as a table name, you can change the actual tables where the names are entered or referenced.

DEFINE/CHARACTERISTIC

Assigns a numeric value to a queue characteristic. The /CHARACTERISTIC qualifier is required. If a value has been assigned to the characteristic, you must delete and redefine the characteristic to alter the assignment of the existing characteristic.

Requires OPER (operator) privilege.

Note

You cannot define more than one characteristic name to a number.

Format

DEFINE/CHARACTERISTIC characteristic-name characteristic-number

Parameters

characteristic-name

Assigns a name to the characteristic being defined. The characteristic name can be the name of an existing characteristic or a string of 1 to 31 characters that defines a new characteristic. The character string can include any uppercase and lowercase letters, digits, the dollar sign ($), and the underscore (_), and must include at least one alphabetic character. Only one characteristic name can be defined to each number.

characteristic-number

Assigns a number in the range 0 to 127 to the characteristic being defined.

Description

The system manager or operator uses the DEFINE/CHARACTERISTIC command to assign a name and number to a particular characteristic for queues in the system. Characteristics can refer to any attribute of a print or batch job that is meaningful for your environment. The name and number of a characteristic are arbitrary, but they must be unique for that characteristic.

Note

Prior to OpenVMS Version 6.0, the DEFINE/CHARACTERISTIC command allowed you to define more than one characteristic name to a number, although this capability was unsupported.

The DEFINE/CHARACTERISTIC command no longer allows you to define more than one characteristic name to a number. However, if your queue configuration requires you to have more than one characteristic name for a single number, you can define logical names to achieve the same result. For example, you might enter the following commands:

```
$ DEFINE/CHARACTERISTIC SECOND_FLOOR 2
$ DEFINE/SYSTEM/EXECUTIVE_MODE SALES_FLOOR SECOND_FLOOR
$ DEFINE/SYSTEM/EXECUTIVE_MODE SALES_DEPT SECOND_FLOOR
```

In this example, the characteristic name SECOND_FLOOR is assigned to the characteristic number 2. The logical names SALES_FLOOR

and SALES_DEPT are then defined as equivalent to the characteristic name SECOND_FLOOR. As a result, the logical names SALES_FLOOR and SALES_DEPT are each equivalent to the characteristic name SECOND_FLOOR and the characteristic number 2. These logical names can be specified as the *characteristic-name* value for any /CHARACTERISTIC=*characteristic-name* qualifier.

In a VMScluster environment, you must define the logical names on every node that requires them.

After characteristics have been defined, they can be associated with print or batch jobs and execution queues. For information on specifying characteristics with jobs, see the description of the /CHARACTERISTICS qualifier of the PRINT and SUBMIT commands.

To find out what characteristics are currently defined for the system, use the SHOW QUEUE/CHARACTERISTICS command. To find out which characteristics have been specified for a particular queue, use the SHOW QUEUE/FULL command. For information on associating characteristics with queues, see the descriptions of the /CHARACTERISTICS qualifier of the INITIALIZE/QUEUE, SET QUEUE, and START/QUEUE commands.

The DELETE/CHARACTERISTIC command deletes a previously defined characteristic.

For more information on specifying queue characteristics, see the *OpenVMS System Manager's Manual*.

Example

```
$ DEFINE/CHARACTERISTIC REDINK 3
```

The DEFINE/CHARACTERISTIC command in this example defines the characteristic REDINK with the number 3. When a user enters the command PRINT/CHARACTERISTICS=REDINK (or PRINT /CHARACTERISTICS=3), the job is printed only if the printer queue has been established with the REDINK or 3 characteristic.

DEFINE/FORM

Assigns a numeric value and attributes to a print form name. The /FORM qualifier is required. To modify a form's name or number, you must delete and redefine the form. Values for any DEFINE/FORM qualifier can be modified by re-entering the DEFINE/FORM command with different values, as long as the form name and number remain the same.

Requires OPER (operator) privilege.

Format

DEFINE/FORM form-name form-number

Parameters

form-name

Assigns a name to the form being defined. The form name can be the name of an existing form type or a string of 1 to 31 characters that defines a new form type. The character string can include any uppercase and lowercase letters, digits, the dollar sign ($), and the underscore (_), and must include at least one alphabetic character.

form-number

Assigns a number in the range 0 to 9999 to the form being defined. The DEFAULT form, which is defined automatically when the system is bootstrapped, is assigned number zero.

Description

The system manager or operator uses the DEFINE/FORM command to assign a name and number to a type of paper stock and printing area for use with printer or terminal queues. When a new queue file is created, the system defines a form named DEFAULT with a form number of zero and all the default attributes.

Some DEFINE/FORM qualifiers specify the area for printing. The LEFT and RIGHT options of the /MARGIN qualifier and the /WIDTH qualifier determine the number of characters per line. Using the RIGHT option of the MARGIN qualifier and the /WIDTH qualifier, you can affect the point at which lines of text wrap. (You cannot use the LEFT and RIGHT options of the /MARGIN qualifier and the /WIDTH qualifier for filling or formatting the text, however.)

You also can use the DEFINE/FORM command to specify different types of paper stock. The /DESCRIPTION qualifier enables you to describe more fully the form name.

After forms have been defined, they can be associated with print jobs and output execution queues. For information on specifying forms with jobs, see the description of the PRINT/FORM command.

To find out what forms have been defined for the system, use the SHOW QUEUE /FORM command. To find out which form is mounted currently on a particular queue and which form is specified as that queue's default form, use the SHOW QUEUE/FULL command. For information on associating forms with queues, see the descriptions of the /DEFAULT and /FORM_MOUNTED qualifiers of the INITIALIZE/QUEUE, SET QUEUE, and START/QUEUE commands.

For more information on how to use forms to control print jobs, see the *OpenVMS System Manager's Manual*.

Qualifiers

/DESCRIPTION=string

A string of up to 255 characters used to provide operator information about the form. The default string is the specified form name.

The string can be used to define the form type more specifically. For example, if you have form names such as LETTER1, LETTER2, and LETTER3, the /DESCRIPTION qualifier could be used to let the users and operators know that LETTER1 refers to the standard corporate letterhead paper (8.5 inches x 11 inches), LETTER2 refers to the smaller corporate letterhead paper (6 inches x 9 inches), and LETTER3 refers to the president's personalized letterhead paper.

Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (‘‘ ’’).

/LENGTH=n

Specifies the physical length of a form page in lines. The default page length is 66 lines, which assumes a standard page length of 11 inches with 6 lines of print per inch. The parameter *n* must be a positive integer greater than zero and not more than 255.

The print symbiont sets the page length of the device equal to the form length. This enables the driver to compute the number of line feeds for devices lacking mechanical form feed.

/MARGIN=(option[,...])

Specifies one or more of the four margin options: BOTTOM, LEFT, RIGHT, and TOP.

BOTTOM=n — Specifies the number of blank lines between the end of the print image area and the end of the physical page; the value of *n* must be between 0 and the value of the /LENGTH qualifier. The default value is 6, which generally means a 1-inch bottom margin.

LEFT=n — Specifies the number of blank columns between the leftmost printing position and the print image area; the value of *n* must be between 0 and the value of the /WIDTH qualifier. The default is 0, which means that the print image area starts as far to the left of the paper as the printer can go.

RIGHT=n — Specifies the number of blank columns between the /WIDTH qualifier and the image area; the value of *n* must be between 0 and the value of the /WIDTH qualifier. When determining the value of the RIGHT option, start at the /WIDTH value and count to the left. The default value is 0, which means that the print image extends as far to the right as the /WIDTH value.

TOP=n — Specifies the number of blank lines between the top of the physical page and the top of the print image; the value of *n* must be between 0 and the value of the /LENGTH qualifier. The default value is 0, which generally means that there is no top margin.

/PAGE_SETUP=(module[,...])
/NOPAGE_SETUP (default)
Specifies one or more modules that set up the device at the start of each page. The modules are located in the device control library. While the form is mounted, the system extracts the specified module and copies it to the printer before each page is printed.

/SETUP=(module[,...])
Specifies one or more modules that set up the device at the start of each file. The modules are located in the device control library. While the form is mounted, the system extracts the specified module and copies it to the printer before each file is printed.

For more information on device control modules, see the chapter on Batch and Print Operations in the *OpenVMS System Manager's Manual*.

/SHEET_FEED
/NOSHEET_FEED (default)
Specifies that print jobs pause at the end of every physical page so that a new sheet of paper can be inserted.

/STOCK=string
Specifies the type of paper stock to be associated with the form. The *string* parameter can be a string of 1 to 31 characters, including the dollar sign, underscore, and all alphanumeric characters. If you specify the /STOCK qualifier you must specify the name of the stock to be associated with the form. If you do not specify the /STOCK qualifier, the name of the stock will be the same as the name of the form.

You can create any string that you want. However, when you are creating forms with the same stock, be sure that the /STOCK string is identical in all the DEFINE/FORM commands that refer to the same type of paper.

If you are defining a number of forms to provide different formatting options, specify the same stock type for each form. Jobs that request any of these forms will print on the same queue. If you want to modify the stock string associated with a form, you can do this only if the form is not referenced by any job or queue.

/TRUNCATE (default)
/NOTRUNCATE
Discards any characters that exceed the current line length (specified by the /WIDTH and /MARGIN=RIGHT qualifiers). The /TRUNCATE qualifier is incompatible with the /WRAP qualifier. If you specify both the /NOTRUNCATE and /NOWRAP qualifiers, the printer prints as many characters on a line as possible. This combination of qualifiers is useful for some types of graphics output.

/WIDTH=n
Specifies the physical width of the paper in terms of columns or character positions. The parameter *n* must be an integer from 0 to 65,535; the default value is 132.

Any lines exceeding this value wrap if the /WRAP qualifier is in effect or are truncated if the /TRUNCATE qualifier is in effect. (If both the /NOTRUNCATE and /NOWRAP qualifiers are in effect, lines print as far as possible.)

The /MARGIN=RIGHT qualifier overrides the /WIDTH qualifier when determining when to wrap lines of text.

/WRAP
/NOWRAP (default)
Causes lines that exceed the current line length (specified by the /WIDTH and /MARGIN=RIGHT qualifiers) to wrap onto the next line. The /WRAP qualifier is incompatible with the /TRUNCATE qualifier. If you specify both the /NOWRAP and /NOTRUNCATE qualifiers, the printer prints as many characters on a line as possible. This combination of qualifiers is useful for some types of graphics output.

Example

```
$ DEFINE/FORM /MARGIN=(TOP=6,LEFT=10) CENTER 3
```

The DEFINE/FORM command in this example defines the form CENTER to have a top margin of 6 and a left margin of 10. The defaults remain in effect for both bottom margin (6) and right margin (0). The form is assigned the number 3.

DEFINE/KEY

Associates an equivalence string and a set of attributes with a key on the terminal keyboard.

Format

DEFINE/KEY key-name equivalence-string

Parameters

key-name

Specifies the name of the key that you are defining. All definable keys on VT52 terminals are located on the numeric keypad. On VT100-series terminals, you can define the left and right arrow keys as well as all the keys on the numeric keypad. On terminals with LK201 keyboards, the following three types of keys can be defined:

- Keys on the numeric keypad
- Keys on the editing keypad (except the up and down arrow keys)
- Keys on the function key row across the top of the keyboard (except keys F1 to F5)

The following table lists the key names in column one. The remaining three columns indicate the key designations on the keyboards of the three different types of terminals that allow key definitions.

Key Name	LK201	VT100-Series	VT52
PF1	PF1	PF1	[blue]
PF2	PF2	PF2	[red]
PF3	PF3	PF3	[gray]
PF4	PF4	PF4	- -
KP0, KP1, ..., KP9	0, 1, ..., 9	0, 1, ..., 9	0, 1, ..., 9
Period	.	.	.
Comma	,	,	n/a
Minus	-	-	n/a
Enter	Enter	ENTER	ENTER
Left	←	←	←
Right	→	→	→
Find (E1)	Find	—	—
Insert Here (E2)	Insert Here	—	—
Remove (E3)	Remove	—	—
Select (E4)	Select	—	—
Prev Screen (E5)	Prev Screen	—	—

Key Name	LK201	VT100-Series	VT52
Next Screen (E6)	Next Screen	—	—
Help	Help	—	—
Do	Do	—	—
F6, F7, ..., F20	F6, F7, ..., F20	—	—

Some definable keys are enabled for definition all the time. Others, including KP0 to KP9, Period, Comma, and Minus, must be enabled for definition purposes. You must enter either the SET TERMINAL/APPLICATION or the SET TERMINAL /NONUMERIC command before using these keys.

On LK201 keyboards, you cannot define the up and down arrow keys or function keys F1 to F5. The left and right arrow keys and the F6 to F14 keys are reserved for command line editing. You must enter the SET TERMINAL/NOLINE_ EDITING command before defining these keys. You can also press Ctrl/V to enable keys F7 to F14. Note that Ctrl/V will not enable the F6 key.

equivalence-string

Specifies the character string to be processed when you press the key. Enclose the string in quotation marks (‘‘ ’’) to preserve spaces and lowercase characters.

Description

The DEFINE/KEY command enables you to assign definitions to the peripheral keys on certain terminals. The terminals include VT52s, the VT100 series, and terminals with LK201 keyboards.

To define keys on the numeric keypads of these terminals, you must first enter the SET TERMINAL/APPLICATION or SET TERMINAL/NONUMERIC command. When your terminal has this setting, the system interprets the keystrokes from keypad keys differently. For example, with SET TERMINAL/NONUMERIC in effect, pressing the 1 key on the keypad does not send the character ‘‘1’’ to the system.

The equivalence string definition can contain different types of information. Definitions often consist of DCL commands. For example, you can assign SHOW TIME to the zero key. When you press 0, the system displays the current date and time. Other definitions can consist of text strings to be appended to command lines. When you define a key to insert a text string, use the /NOTERMINATE qualifier so that you can continue typing more data after the string has been inserted.

In most instances you will want to use the echo feature. The default setting is /ECHO. With /ECHO set, the key definition is displayed on the screen each time you press the key.

You can use the /STATE qualifier to increase the number of key definitions available on your terminal. The same key can be assigned any number of definitions, as long as each definition is associated with a different state. State names can contain any alphanumeric characters, dollar signs, and underscores. Be sure to create a state name that is easy to remember and type and, if possible, one that might remind you of the types of definitions you created for that state. For example, you can create a state called SETSHOW. The key definitions for this state might all refer to various DCL SET and SHOW commands. If you are used to the EDT Editor, you might define a state as GOLD. Then, using the /IF_STATE

qualifier, you can assign different definitions to keys used in combination with a key defined as GOLD.

The SET KEY command changes the keypad state. Use the SHOW KEY command to display key definitions and states.

Qualifiers

/ECHO (default)
/NOECHO
Displays the equivalence string on your screen after the key has been pressed. You cannot use the /NOECHO qualifier with the /NOTERMINATE qualifier.

/ERASE
/NOERASE (default)
Determines whether the current line is erased before the key translation is inserted.

/IF_STATE=(state-name,...)
/NOIF_STATE
Specifies a list of one or more states, one of which must be in effect for the key definition to work. The /NOIF_STATE qualifier has the same meaning as /IF_STATE=current_state. The state name is an alphanumeric string. States are established with the /SET_STATE qualifier or the SET KEY command. If you specify only one state name, you can omit the parentheses. By including several state names, you can define a key to have the same function in all the specified states.

/LOCK_STATE
/NOLOCK_STATE (default)
Specifies that the state set by the /SET_STATE qualifier remain in effect until explicitly changed. (By default, the /SET_STATE qualifier is in effect only for the next definable key you press or the next read-terminating character that you type.) This qualifier can be specified only with the /SET_STATE qualifier.

/LOG (default)
/NOLOG
Displays a message indicating that the key definition has been successfully created.

/SET_STATE=state-name
/NOSET_STATE (default)
Causes the specified state-name to be set when the key is pressed. (By default, the current locked state is reset when the key is pressed.) If you have not included this qualifier with a key definition, you can use the SET KEY command to change the current state. The state name can be any alphanumeric string; specify the state as a character string enclosed in quotation marks.

/TERMINATE
/NOTERMINATE (default)
Specifies whether the current equivalence string is to be processed immediately when the key is pressed (equivalent to entering the string and pressing the Return key). By default, you can press other keys before the definition is processed. This allows you to create key definitions that insert text into command lines, after prompts, or into other text that you are entering.

Examples

1.
```
$ DEFINE/KEY PF3 "SHOW TIME" /TERMINATE
%DCL-I-DEFKEY, DEFAULT key PF3 has been defined
$ PF3
$ SHOW TIME
  14-DEC-1994 14:43:59
```

In this example, the DEFINE/KEY command defines the PF3 key on the keypad to perform the SHOW TIME command. DEFAULT refers to the default state.

2.
```
$ DEFINE/KEY PF1 "SHOW " /SET_STATE=GOLD/NOTERMINATE/ECHO
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
$ DEFINE/KEY PF1 " DEFAULT" /TERMINATE/IF_STATE=GOLD/ECHO
%DCL-I-DEFKEY, GOLD key PF1 has been defined
$ PF1
$ PF1
$ SHOW DEFAULT
DISK1:[JOHN.TEST]
```

In this example, the first DEFINE/KEY command defines the PF1 key to be the string SHOW. The state is set to GOLD for the subsequent key. The /NOTERMINATE qualifier instructs the system not to process the string when the key is pressed. The second DEFINE/KEY command defines the use of the PF1 key when the keypad is in the GOLD state. When the keypad is in the GOLD state, pressing PF1 causes the current read to be terminated.

If you press the PF1 key twice, the system displays and processes the SHOW DEFAULT command.

The word DEFAULT in the second line of the example indicates that the PF1 key has been defined in the default state. Note the space before the word DEFAULT in the second DEFINE/KEY command. If the space is omitted, the system fails to recognize DEFAULT as the keyword for the SHOW command.

3.
```
$ SET KEY/STATE=ONE
%DCL-I-SETKEY, keypad state has been set to ONE
$ DEFINE/KEY PF1 "ONE"
%DCL-I-DEFKEY, ONE key PF1 has been defined
$ DEFINE/KEY/IF_STATE=ONE PF1 "ONE"
%DCL-I-DEFKEY, ONE key PF1 has been defined
```

This example shows two ways to define the PF1 key to be ‘‘ONE’’ for state ONE.

The second DEFINE/KEY command shows the preferred method for defining keys. This method eliminates the possibility of error by specifying the state in the same command as the key definition.

DELETE

Deletes one or more files from a mass storage disk volume.

Requires delete (D) access to the file and write (W) access to the parent directory. If the target file is itself a directory, the directory must be empty.

Format

DELETE filespec[,...]

Parameter

filespec[,...]

Specifies the names of one or more files to be deleted from a mass storage disk volume. The first file specification must contain an explicit or default directory specification plus an explicit file name, file type, and version number. Subsequent file specifications need contain only a version number; the defaults will come from the preceding specification. The asterisk (*) and the percent sign (%) wildcard characters can be used in any of the file specification fields.

If you omit the directory specification or device name, the current default device and directory are assumed.

If the file specification contains a null version number (a semicolon (;) followed by no file version number), a version number of 0, or one or more spaces in the version number, the latest version of the file is deleted.

To delete more than one file, separate the file specifications with either commas (,) or plus signs (+).

Qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /BACKUP qualifier selects files according to the dates of their most recent backups. This qualifier is incompatible with the /CREATED, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /BEFORE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

Specify the UIC by using standard UIC format as described in the *OpenVMS Guide to System Security*.

/CONFIRM
/NOCONFIRM (default)
Controls whether a request is issued before each delete operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	Ctrl/Z
1	0	ALL
	Return	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers include: NO, FALSE, 0, and pressing the Return key. Entering QUIT or pressing Ctrl/Z indicates that you want to stop processing the command at that point. When you respond by entering ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the /BACKUP, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/ERASE
/NOERASE (default)
When you delete a file, the area in which the file was stored is returned to the system for future use. The data that was stored in that location still exists in the system until new data is written over it. When you specify the /ERASE qualifier, the storage location is overwritten with a system specified pattern so that the data no longer exists.

/EXCLUDE=(filespec[,...])
Excludes the specified files from the delete operation. You can include a directory but not a device in the file specification. The asterisk (*) and the percent sign (%) wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /EXPIRED qualifier selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the /BACKUP, /CREATED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/LOG
/NOLOG (default)
Controls whether the DELETE command displays the file specification of each file after its deletion.

/MODIFIED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /MODIFIED qualifier selects files according to the dates on which they were last modified. This qualifier is incompatible with the /BACKUP, /CREATED, and /EXPIRED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time modifiers, the default is the /CREATED qualifier.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /SINCE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

Examples

1.
```
$ DELETE   COMMON.SUM;2
```

   The DELETE command deletes the file COMMON.SUM;2 from the current default disk and directory.

2.
```
$ DELETE *.OLD;*
```

   The DELETE command deletes all versions of files with file type .OLD from the default disk directory.

3.
```
$ DELETE  ALPHA.TXT;*, BETA;*, GAMMA;*
```

   The DELETE command deletes all versions of the files ALPHA.TXT, BETA.TXT, and GAMMA.TXT. The command uses the file type of the first input file as a temporary default. Note, however, that some form of version number (here specified as the asterisk (*) wildcards) must be included in each file specification.

4.
```
$ DELETE /BEFORE=15-APR/LOG *.DAT;*
%DELETE-I-FILDEL, DISK2:[MALCOLM]ASSIGN.DAT;1 deleted (5 block)
%DELETE-I-FILDEL, DISK2:[MALCOLM]BATCHAVE.DAT;3 deleted (4 blocks)
%DELETE-I-FILDEL, DISK2:[MALCOLM]BATCHAVE.DAT;2 deleted (4 blocks)
%DELETE-I-FILDEL, DISK2:[MALCOLM]BATCHAVE.DAT;1 deleted (4 blocks)
%DELETE-I-FILDEL, DISK2:[MALCOLM]CANCEL.DAT;1 deleted (2 blocks)
%DELETE-I-FILDEL, DISK2:[MALCOLM]DEFINE.DAT;1 deleted (3 blocks)
%DELETE-I-FILDEL, DISK2:[MALCOLM]EXIT.DAT;1 deleted (1 block)
%DELETE-I-TOTAL, 7 files deleted (23 blocks)
```

   The DELETE command deletes all versions of all files with file type .DAT that were either created or updated before April 15 of this year. The /LOG qualifier not only displays the name of each file deleted, but also the total number of files deleted.

5.
```
$ DELETE A.B;
```

The DELETE command deletes the file A.B with the highest version number.

6.
```
$ DELETE/CONFIRM/SINCE=TODAY [MALCOLM.TESTFILES]*.OBJ;*
DISK0:[MALCOLM.TESTFILES]AVERAG.OBJ;1, delete? [N]:Y
DISK0:[MALCOLM.TESTFILES]SCANLINE.OBJ;4, delete? [N]:N
DISK0:[MALCOLM.TESTFILES]SCANLINE.OBJ;3, delete? [N]:N
DISK0:[MALCOLM.TESTFILES]SCANLINE.OBJ;2, delete? [N]:N
DISK0:[MALCOLM.TESTFILES]WEATHER.OBJ;3, delete? [N]:Y
```

The DELETE command examines all versions of files with file type OBJ in the subdirectory [MALCOLM.TESTFILES], and locates those that were created or modified today. Before deleting each file, it requests confirmation that the file should be deleted. The default response—N—is given in brackets.

7.
```
$ DIRECTORY [.SUBTEST]
%DIRECT-W-NOFILES, no files found
$ SET PROTECTION SUBTEST.DIR/PROTECTION=OWNER:D
$ DELETE SUBTEST.DIR;1
```

Before the directory file SUBTEST.DIR is deleted, the DIRECTORY command is used to verify that there are no files cataloged in the directory. The SET PROTECTION command redefines the protection for the directory file so that it can be deleted; then the DELETE command deletes it.

8.
```
$ DELETE DALLAS"THOMAS SECRET"::DISK0:[000,000]DECODE.LIS;1
```

This DELETE command deletes the file DECODE.LIS;1 from the directory [000,000] on device DISK0 at remote node DALLAS. The user name and password follow the remote node name.

9.
```
$ DELETE NODE12::"DISK1:DEAL.BIG"
$ DELETE NODE12::DISK1:DEAL.BIG;
```

Either of these DELETE commands can be used to delete the file DEAL.BIG on device ZZZ1 at remote node NODE12. Note that the DELETE command requires an explicit version number in a file specification, but the file to be deleted is on a remote node whose file syntax does not recognize version numbers. (NODE12 is an RT–11 node.) Therefore, the file specification must either be enclosed in quotation marks (‘‘ ’’) or entered with a null version number (that is, a trailing semicolon [;]).

DELETE/CHARACTERISTIC

Deletes the definition of a queue characteristic previously established with the DEFINE/CHARACTERISTIC command. The /CHARACTERISTIC qualifier is required.

Requires OPER (operator) privilege.

Format

DELETE/CHARACTERISTIC characteristic-name

Parameter

characteristic-name
Specifies the name of the characteristic to be deleted.

Description

The DELETE/CHARACTERISTIC command deletes a characteristic from the system characteristic table.

To modify a characteristic's name or number, you must delete and redefine the characteristic.

Qualifier

/LOG
/NOLOG (default)
Controls whether the DELETE/CHARACTERISTIC command displays the name of each characteristic after its deletion.

Example

```
$ DEFINE/CHARACTERISTIC BLUE 7
   .
   .
   .
$ DELETE/CHARACTERISTIC BLUE
$ DEFINE/CHARACTERISTIC BLUE_INK 7
```

The DEFINE/CHARACTERISTIC command in this example establishes the characteristic BLUE, with number 7, to mean blue ink ribbons for printers. To change the name of the characteristic, enter the DELETE/CHARACTERISTIC command. Then enter another DEFINE/CHARACTERISTIC command to rename the characteristic to BLUE_INK, using the characteristic number 7.

DELETE/ENTRY

Deletes one or more print or batch jobs. The jobs can be in progress or waiting in the queue. The /ENTRY qualifier is required.

Requires manage (M) access to the queue, or delete (D) access to the specified jobs.

Format

DELETE/ENTRY=(entry-number[,...]) [queue-name[:]]

Parameters

entry-number[,...]
Specifies the entry number (or a list of entry numbers) of jobs to be deleted. If you specify only one entry number, you can omit the parentheses. If you do not specify a queue name, you can delete entries from multiple queues.

The system assigns a unique entry number to each queued print or batch job in the system. By default, the PRINT and SUBMIT commands display the entry number when they successfully queue a job for processing. These commands also create or update the local symbol $ENTRY to reflect the entry number of the most recently queued job. To find a job's entry number, enter the SHOW ENTRY or SHOW QUEUE command.

queue-name[:]
Specifies the name of the queue where the jobs are located. The queue name can refer either to the queue to which the job was submitted or to the queue where the job is executing. The *queue-name* parameter is optional syntax. However, when you specify a queue name, the operating system uses it to verify an entry in the specific queue before deleting the entry.

Description

The DELETE/ENTRY command deletes one or more jobs from a queue. If you specify a queue name and more than one entry number with a DELETE/ENTRY command, all the jobs must be located in the same queue.

You can delete jobs that are currently executing, as well as jobs that are in other states. For example, DELETE/ENTRY can delete a job that is currently in a holding or a pending state.

Qualifier

/LOG
/NOLOG (default)
Controls whether the DELETE/ENTRY command displays the entry number of each batch or print job that it deletes.

Examples

1.
```
$ PRINT/HOLD   ALPHA.TXT
Job ALPHA (queue SYS$PRINT, entry 110) holding
   .
   .
   .
$ DELETE/ENTRY=110  SYS$PRINT
```

The PRINT command in this example queues a copy of the file ALPHA.TXT in a HOLD status, to defer its printing until a SET ENTRY/RELEASE command is entered. The system displays the job name, the entry number, the name of the queue in which the job was entered, and the status. Later, the DELETE/ENTRY command requests that the entry be deleted from the queue SYS$PRINT.

2.
```
$ SUBMIT/AFTER=18:00  WEATHER
Job WEATHER (queue SYS$BATCH, entry 203) holding until 14-DEC-1994
18:00
$ SUBMIT/HOLD/PARAMETERS=SCANLINE  DOFOR
Job DOFOR (queue SYS$BATCH, entry 210) holding
   .
   .
   .
$ DELETE/ENTRY=(203,210)/LOG
%DELETE-W-SEARCHFAIL, error searching for 203
-JBC-E-NOSUCHENT, no such entry
%DELETE-I-DELETED, entry 210 aborting or deleted
```

The SUBMIT commands in this example queue the command procedures WEATHER.COM and DOFOR.COM for processing as batch jobs. WEATHER.COM is queued for execution after 6:00 P.M. DOFOR.COM is queued in a HOLD status and cannot execute until you enter a SET ENTRY /RELEASE command. Later, the DELETE/ENTRY/LOG command requests that the system delete both these entries from the queue and display a message indicating that the entries have been deleted.

The job WEATHER (entry 203) has completed by the time the DELETE /ENTRY/LOG command is entered. Thus, entry 203 no longer exists. Note that a message indicates that there is no entry 203 in the queue. The job DOFOR (entry 210) is in a HOLD status when the DELETE/ENTRY/LOG command is entered. Thus, the system deletes entry 210 from the queue and displays a message to that effect.

3.
```
$ PRINT CHAPTER8.MEM
Job CHAPTER8 (queue SYS$PRINT, entry 25) pending on queue SYS$PRINT
   .
   .
   .
$ SHOW QUEUE SYS$PRINT
Printer queue SYS$PRINT, on PARROT::PARROT$LPA0, mounted form DEFAULT

Entry  Jobname         Username             Status
-----  -------         --------             ------
   24  CHAPTER7        SMITH                Pending
   25  CHAPTER8        SMITH                Pending
$ DELETE/ENTRY=25
```

The PRINT command in this example submits the file CHAPTER8.MEM to the printer queue SYS$PRINT. Later, user Smith needs to edit the file again before printing it. Using the SHOW QUEUE command, Smith verifies that the job is still pending and that the entry number for the job is 25. Smith then enters the DELETE/ENTRY command to delete the job from the queue.

DELETE/FORM

Deletes a form (for printer or terminal queues) previously established with the DEFINE/FORM command. The /FORM qualifier is required.

Requires OPER (operator) privilege.

Format

DELETE/FORM form-name

Parameter

form-name
Specifies the name of the form to be deleted.

Description

The DELETE/FORM command deletes a form definition from the system forms table. When you delete a form, there can be no outstanding references to the form either in queues that have been mounted with the form or by jobs requesting that form. To locate all references to the form, use the SHOW QUEUE/FULL command.

To modify a form's name or number, you must delete and redefine the form. Values for any DEFINE/FORM qualifier can be modified by re-entering the DEFINE/FORM command with different values, as long as the form name and number remain the same.

Qualifier

/LOG
/NOLOG (default)
Controls whether the DELETE/FORM command displays the name of each form after its deletion.

Examples

1.
```
$ DELETE/FORM CENTER
```

The DELETE/FORM command in this example deletes the form named CENTER.

2.
```
$ DEFINE/FORM -
_$ /DESCRIPTION="letter size continuous form paper" CFLET 7
    .
    .
    .
$ DELETE/FORM CFLET
$ DEFINE/FORM -
_$ /DESCRIPTION="letter size continuous form paper" LETTER_CONT 7
```

The DEFINE/FORM command in this example establishes the form CFLET with number 7 to mean continuous-form paper 8.5 inches by 11 inches. To change the name of the form, delete the form named CFLET and define a new one named LETTER_CONT.

DELETE/INTRUSION_RECORD

Removes an entry from the break-in database.

Requires CMKRNL (change mode to kernel) and SECURITY privileges.

Format

DELETE/INTRUSION_RECORD source

Parameter

source
Specifies the name of the device or the remote system where the user is attempting to log in. The source name can be presented in the syntax of another operating system domain, for example, one that is case sensitive or conflicts with DCL syntax rules. In such cases, you must enclose the source parameter in quotation marks.

Description

Use the DELETE/INTRUSION_RECORD command to remove an entry from the break-in database. For example, if the user Hammer repeatedly attempted to log in to terminal TTA24 with an expired password, the SHOW INTRUSION command would display the following entry:

```
Intrusion  Type      Count    Expiration   Source

TERM_USER   INTRUDER    9      10:29:39.16   TTA24:HAMMER
```

The terminal is locked out of the system because the login failure limit has been reached. When Hammer approaches you and you identify the problem as an expired password, you can then use the DELETE/INTRUSION command to remove the record from the break-in database.

Examples

1. `$ DELETE/INTRUSION_RECORD TTC2:`

   In this example, the DELETE/INTRUSION_RECORD command removes all intrusion records generated by break-in attempts on TTC2. No user name is specified because none of the login failures occurred for valid users.

2. `$ DELETE/INTRUSION_RECORD "AV34C2/LC-2-10":FORGETFUL`

   In this example, the source of the break-in is a local terminal that is connected to a terminal server. To delete the record from the break-in database, you must enclose the terminal port name within quotation marks so that the operating system interprets the slash as a foreign character and not as a qualifier.

3. `$ DELETE/INTRUSION_RECORD NODE1::HAMMER`

   This command removes all intrusion entries generated from node NODE1 for user HAMMER.

DELETE/KEY

Deletes key definitions that have been established by the DEFINE/KEY command. The /KEY qualifier is required.

Format

DELETE/KEY [key-name]

Parameter

key-name
Specifies the name of the key to be deleted. This parameter is incompatible with the /ALL qualifier.

Qualifiers

/ALL
Deletes all key definitions in the specified state; the default is the current state. If you use the /ALL qualifier, do not specify a key name. Use the /STATE qualifier to specify one or more states.

/LOG (default)
/NOLOG
Controls whether messages are displayed indicating that the specified key definitions have been deleted.

/STATE=(state-name[,...])
/NOSTATE (default)
Specifies the name of the state for which the specified key definition is to be deleted. The default state is the current state. If you specify only one state name, you can omit the parentheses. State names can be any alphanumeric string.

Examples

1.
```
$ DELETE/KEY/ALL
%DCL-I-DELKEY, DEFAULT key PF1 has been deleted
%DCL-I-DELKEY, DEFAULT key PF2 has been deleted
%DCL-I-DELKEY, DEFAULT key PF3 has been deleted
%DCL-I-DELKEY, DEFAULT key PF4 has been deleted
$
```

In this example, the user has defined keys PF1 to PF4 in the default state. The DELETE/KEY command deletes all key definitions in the current state, which is the default state.

2.
```
$ DEFINE/KEY PF3 "SHOW TIME" /TERMINATE
%DCL-I-DEFKEY, DEFAULT key PF3 has been defined
$ PF3
$ SHOW TIME
 14-DEC-1994 14:43:59
    .
    .
    .
$ DELETE/KEY PF3
%DCL-I-DELKEY, DEFAULT key PF3 has been deleted
$ PF3
$
```

In this example, the DEFINE/KEY command defines the PF3 key on the keypad as SHOW TIME. To delete the definition for the PF3 key, use the DELETE/KEY command. When the user presses PF3, only the system prompt is displayed.

DELETE/QUEUE

Deletes a print or batch queue created by the INITIALIZE/QUEUE command, and deletes all the jobs in the queue. The /QUEUE qualifier is required.

Requires manage (M) access to the queue.

Format

DELETE/QUEUE queue-name[:]

Parameter

queue-name[:]
Specifies the name of the queue to be deleted.

Description

To delete a queue, use the following procedure:

1. Stop the specified queue by using the STOP/QUEUE/NEXT command.

   The STOP/QUEUE/NEXT command stops the specified queue after all executing jobs have completed processing. Wait for any executing jobs to complete processing.

2. Make sure that there are no outstanding references to the specified queue.

   If a generic queue refers to the specified queue as a target execution queue, you must remove the specified queue from the list of target execution queues.

   If a logical queue refers to the specified queue, you must deassign the logical queue.

   If the specified queue is a generic queue, jobs that were entered initially on the generic queue and still exist on any of its target queues count as references to the specified queue. Before you can delete the specified queue, you must delete any jobs that were submitted originally to the specified queue and are executing on its target queues, or you must wait until these jobs have completed processing.

3. To move jobs from the specified queue to another queue, use the SET ENTRY /REQUEUE or ASSIGN/MERGE commands. Any jobs that remain in the specified queue are deleted when the queue is deleted.

4. Enter the DELETE/QUEUE command.

Qualifier

/LOG
/NOLOG (default)
Controls whether the DELETE/QUEUE command displays the name of each queue after it is deleted.

Example

```
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START/ON=LPA0 LPA0_QUEUE
   .
   .
   .
$ STOP/QUEUE/NEXT LPA0_QUEUE
$ DELETE/QUEUE LPA0_QUEUE
```

In this example, the first command initializes and starts the printer queue LPA0_QUEUE. The STOP/QUEUE/NEXT command stops the queue. The DELETE/QUEUE command deletes the queue.

DELETE/QUEUE/MANAGER

Deletes a queue manager on a node or VAXcluster system. All queues and jobs managed by the specified queue manager are also deleted. You must first stop the queue manager. The /NAME_OF_MANAGER qualifier is required.

Requires OPER (operator) and SYSNAM (system logical name) privileges.

Format

DELETE/QUEUE/MANAGER/NAME_OF_MANAGER=name

Parameter

None.

Description

To delete a queue manager, use the following procedure:

1. Stop the specified queue manager by using the STOP/QUEUE/MANAGER /CLUSTER/NAME_OF_MANAGER=name command.
2. Enter the DELETE/QUEUE/MANAGER/NAME_OF_MANAGER command, specifying the queue manager name.

Qualifier

/NAME_OF_MANAGER=string
Identifies the name of the queue manager to be deleted. The /NAME_OF_ MANAGER qualifier is required. The required name value may be up to 31 characters long and may be a logical name.

Example

```
$ DELETE/QUEUE/MANAGER/NAME_OF_MANAGER=BATCH_MANAGER
```

The DELETE/QUEUE/MANAGER/NAME_OF_MANAGER command in this example deletes the queue manager named BATCH_MANAGER. The command removes all references to the specified queue manager from the shared master file of the queue database and deletes the queue and journal files associated with the BATCH_MANAGER's database.

DELETE/SYMBOL

Deletes one or all symbol definitions from a local or global symbol table. The /SYMBOL qualifier is required.

Format

DELETE/SYMBOL [symbol-name]

Parameter

symbol-name
Specifies the name of the symbol to be deleted. A name is required unless the /ALL qualifier is specified. The *symbol-name* parameter is incompatible with the /ALL qualifier. Symbol names can have from 1 to 255 characters. By default, the DELETE/SYMBOL command assumes that the symbol is in the local symbol table for the current command procedure.

Description

The DELETE/SYMBOL command deletes a symbol definition from a symbol table. If you do not specify either the global or local symbol table, the symbol is deleted from the local table. If you specify both the /GLOBAL and /LOCAL qualifiers, only the last specified qualifier is accepted. The /SYMBOL qualifier must always immediately follow the DELETE command name.

Qualifiers

/ALL
Deletes all symbols from the specified table. If you do not specify either the /LOCAL or the /GLOBAL qualifier, all symbols defined at the current command level are deleted. The /ALL qualifier is incompatible with the *symbol-name* parameter.

/GLOBAL
Deletes the symbol from the global symbol table of the current process.

/LOCAL (default)
Deletes the symbol from the local symbol table of the current process.

/LOG
/NOLOG (default)
Controls whether an informational message listing each symbol being deleted is displayed.

Examples

1.
```
$ DELETE/SYMBOL/ALL
```

   In this example, the DELETE/SYMBOL command deletes all symbol definitions at the current command level.

2.
```
$ DELETE/SYMBOL/LOG KUDOS
%DCL-I-DELSYM, LOCAL symbol KUDOS has been deleted
```

   In this example, the DELETE/SYMBOL command deletes the symbol KUDOS from the local symbol table for the current process. In addition, the /LOG qualifier causes an informational message, listing the symbol being deleted, to be displayed.

3.
```
$ DELETE/SYMBOL/GLOBAL PDEL
```

   In this example, the DELETE/SYMBOL command deletes the symbol named PDEL from the global symbol table for the current process.

DEPOSIT

Replaces the contents of the specified locations in virtual memory and displays the new contents.

The DEPOSIT command, together with the EXAMINE command, aids in debugging programs interactively. The DCL command DEPOSIT is similar to the DEPOSIT command of the OpenVMS Debugger.

Requires user-mode read (R) and write (W) access to the virtual memory location whose contents you wish to change.

Format

DEPOSIT location=data[,...]

Parameters

location

Specifies the starting virtual address or range of virtual addresses (where the second address is larger than the first) whose contents are to be changed. A location can be any valid integer expression containing an integer value, a symbol name, a lexical function, or a combination of these entities. Radix qualifiers determine the radix in which the address is interpreted; hexadecimal is the initial default radix. Symbol names are always interpreted in the radix in which they were defined. The radix operators %X, %D, or %O can precede the location. A hexadecimal value must begin with a number (or be preceded by %X).

The specified location must be within the virtual address space of the image currently running in the process.

The DEPOSIT and EXAMINE commands maintain a pointer to a current memory location. The DEPOSIT command sets this pointer to the byte following the last byte modified; you can refer to this pointer by using a period (.) in subsequent EXAMINE and DEPOSIT commands. If the DEPOSIT command cannot deposit the specified data, the pointer does not change. The EXAMINE command does not change the value of the pointer.

data[,...]

Specifies the data to be deposited into the specified locations. By default, the data is assumed to be in hexadecimal format; it is then converted to binary format and is written into the specified location.

If you specify more than one item, separate the items with commas (,). The DEPOSIT command writes the data in consecutive locations, beginning with the address specified.

When non-ASCII data is deposited, you can specify each item of data using any valid integer expression.

When ASCII data is deposited, only one item of data is allowed. All characters to the right of the equal sign are considered to be part of a single string. The characters are converted to uppercase, and all spaces are compressed.

DEPOSIT

Description

When the DEPOSIT command completes, it displays both the virtual memory address into which data is deposited and the new contents of the location, as follows:

```
address:  contents
```

If the specified address can be read from but not written to by the current access mode, the DEPOSIT command displays the original contents of the location. If the specified address can be neither read from nor written to, the DEPOSIT command displays asterisks (*) in the data field. The DEPOSIT command maintains a pointer at that location (at the byte following the last byte modified).

If you specify a list of numeric values, some but not all of the values may be successfully deposited before an access violation occurs. If an access violation occurs while ASCII data is being deposited, nothing is deposited.

Radix Qualifiers: The radix default for a DEPOSIT or EXAMINE command determines how the command interpreter interprets numeric literals. The initial default radix is hexadecimal; all numeric literals in the command line are assumed to be hexadecimal values. If a radix qualifier modifies the command, that radix becomes the default for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it. For example:

```
$ DEPOSIT/DECIMAL 900=256
00000384:  256
```

The DEPOSIT command interprets both the location 900 and the value 256 as decimal. All subsequent DEPOSIT and EXAMINE commands assume that numbers you enter for addresses and data are decimal. Note that the DEPOSIT command always displays the address location in hexadecimal.

Symbol values defined by = (assignment statement) commands are always interpreted in the radix in which they were defined.

Note that hexadecimal values entered as deposit locations or as data to be deposited must begin with a numeric character (0 to 9). Otherwise, the command interpreter assumes that you have entered a symbol name and attempts symbol substitution.

You can use the radix operators %X, %D, or %O to override the current default when you enter the DEPOSIT command. For example:

```
$ DEPOSIT/DECIMAL %X900=10
```

This command deposits the decimal value 10 in the location specified as hexadecimal 900.

Length Qualifiers: The initial default length unit for the DEPOSIT command is a longword. If a list of data values is specified, the data is deposited into consecutive longwords beginning at the specified location. If a length qualifier modifies the command, that length becomes the default for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it. If you specify data values that are longer than the specified length, an error occurs.

Length qualifiers are ignored when ASCII values are deposited.

Restriction on Placement of Qualifiers: The DEPOSIT command analyzes expressions arithmetically. Therefore, qualifiers, which must be preceded by a slash (/), must appear immediately after the command name to be interpreted correctly.

Qualifiers

/ASCII
Indicates that the specified data is ASCII.

Only one data item is allowed; all characters to the right of the equal sign (=) are considered to be part of a single string. Unless they are enclosed within quotation marks (‘‘ ’’), characters are converted to uppercase and multiple spaces are compressed to a single space before the data is written in memory.

The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory. When you specify /ASCII, or when ASCII mode is the default, the location you specify is assumed to be hexadecimal.

/BYTE
Requests that data be deposited 1 byte at a time.

/DECIMAL
Indicates that the data is decimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/HEXADECIMAL
Indicates that the data is hexadecimal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/LONGWORD
Requests that data be deposited a longword at a time.

/OCTAL
Indicates that the data is octal. The DEPOSIT command converts the data to its binary equivalent before placing it in virtual memory.

/WORD
Requests that the data be deposited one word at a time.

Examples

1.
```
$ RUN MYPROG
    .
    .
    .
Ctrl/Y
$ EXAMINE %D2145876444
7FE779DC:  0000000000
$ DEPOSIT .=17
7FE779DC:  0000000017
$ CONTINUE
```

The RUN command executes the image MYPROG.EXE; subsequently, Ctrl/Y interrupts the program. Assuming that the initial defaults of the /HEXADECIMAL and /LONGWORD qualifiers are in effect, the DEPOSIT command places a longword value 17 (23 decimal) in virtual memory location 2145876444.

Because the EXAMINE command sets up a pointer to the current memory location, which in this case is virtual address 2145876444, you can refer to this location with a period (.) in the DEPOSIT command.

The CONTINUE command resumes execution of the image.

2.
```
$ DEPOSIT/ASCII   2C00=FILE: NAME: TYPE:
00002C00:  FILE: NAME: TYPE:...
```

In this example, the DEPOSIT command deposits character data at hexadecimal location 2C00 and displays the contents of the location after modifying it. Because the current default length is a longword, the response from the DEPOSIT command displays full longwords. The ellipsis (. . .) indicates that the remainder of the last longword of data contains information that was not modified by the DEPOSIT command.

3.
```
$ EXAMINE 9C0             !  Look at Hex location 9C0
000009C0:  8C037DB3
$ DEPOSIT .=0             !  Deposit longword of 0
000009C0:  00000000
$ DEPOSIT/BYTE .=1        !  Put 1 byte at next location
000009C4:  01
$ DEPOSIT .+2=55          !  Deposit 55 next
000009C7:  55
$ DEPOSIT/LONG .=0C,0D,0E !  Deposit longwords
000009C8:  0000000C 0000000D 0000000E
```

The sequence of DEPOSIT commands in the above example illustrates how the DEPOSIT command changes the current position pointer. Note that after you specify the /BYTE qualifier, all data is deposited and displayed in bytes, until the /LONGWORD qualifier restores the system default.

4.
```
$ BASE=%X200               !  Define a base address
$ LIST=BASE+%X40           !  Define offset from base
$ DEPOSIT/DECIMAL LIST=1,22,333,4444
00000240:  00000001 00000022 00000333 00004444
$ EXAMINE/HEX LIST:LIST+0C !  Display results in hex
00000240:  00000001 00000016 0000014D 0000115C
```

The assignment statements define a base address in hexadecimal and a label at a hexadecimal offset from the base address. The DEPOSIT command reads the list of values and deposits each value into a longword, beginning at the specified location. The EXAMINE command requests a hexadecimal display of these values.

DIAGNOSE (Alpha Only)

Alpha

On Alpha, the DIAGNOSE command invokes the DECevent management utility and selectively reports the contents of one or more event log files. For a complete description of the DECevent utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

DIAGNOSE [qualifier...] [filespec][,...] ♦

DIFFERENCES

Compares the contents of two disk files and displays a listing of the records that do not match.

Format

DIFFERENCES input1-filespec [input2-filespec]

Parameters

input1-filespec

Specifies the first file to be compared. The file specification must include a file name and a file type. The asterisk (*) and the percent sign (%) wildcard characters are not allowed.

input2-filespec

Specifies the second file to be compared. Unspecified fields default to the corresponding fields in the *input1-filespec* parameter. The asterisk (*) and the percent sign (%) wildcard characters are not allowed.

If you do not specify a secondary input file, the DIFFERENCES command uses the next lower version of the primary input file.

Description

Use the DIFFERENCES command to determine whether two files are identical and, if not, how they differ. The DIFFERENCES command compares the two specified files on a record-by-record basis and produces an output file that lists the DIFFERENCES, if any.

The qualifiers for the DIFFERENCES command can be categorized according to their functions, as follows:

- Qualifiers that request the DIFFERENCES command to ignore data in each record:

  /COMMENT_DELIMITERS
  /IGNORE

  These qualifiers allow you to define characters that denote comments or to designate characters or classes of characters to ignore when comparing files. For example, you can have the DIFFERENCES command ignore extra blank lines or extra spaces within lines.

  By default, the DIFFERENCES command compares every character in each record.

- Qualifiers that control the format of the information contained in the list of differences:

  /CHANGE_BAR
  /IGNORE
  /MERGED
  /MODE
  /PARALLEL
  /SEPARATED
  /SLP
  /WIDTH

By default, the DIFFERENCES command merges the differences it finds in the files being compared. It lists each record in the file that has no match in the other input file and then lists the next record that it finds that does have a match.

By default, the DIFFERENCES command also supplies a line number with each listed record, and it lists the records with all designated ignore characters deleted.

You can specify combinations of qualifiers to request an output listing that includes the comparison in more than one format. Note that SLP output is incompatible with all other types of output; parallel output can be generated only in ASCII mode.

- Qualifiers that control the extent of the comparison:

  /MATCH
  /MAXIMUM_DIFFERENCES
  /WINDOW

  By default, the DIFFERENCES command reads every record in the master input file and looks for a matching record in the revision input file. A search for a match between the two input files continues until either a match is found or the ends of the two files are reached. Sections of the two files are considered a match only if three sequential records are found to be identical in each file.

By default, DIFFERENCES command output is written to the current SYS$OUTPUT device. Use the /OUTPUT qualifier to request that the DIFFERENCES command write the output to an alternate file or device.

The DIFFERENCES command terminates with an exit status. The following severity levels indicate the result of the comparison:

SUCCESS	Files are identical.
INFORMATIONAL	Files are different.
WARNING	User-specified maximum number of DIFFERENCES has been exceeded.
ERROR	Insufficient virtual memory to complete comparison.

All severity levels other than SUCCESS indicate that the two input files are different.

Qualifiers

/CHANGE_BAR[=([change-char][,[NO]NUMBER])]

Marks differences using the specified character. The /CHANGE_BAR qualifier displays output that depends on where the qualifier is placed. The following examples describe the result of /CHANGE_BAR qualifier placement.

The following placement displays the latest version of *input.file* with the pound sign (#) preceding any lines that differ from the preceding version of *input.file*:

```
$ DIFFERENCES input.file/CHANGE_BAR=#
```

The following placement displays *input.file;2* with the pound sign (#) preceding any lines that differ from *input.file;1*:

```
$ DIFFERENCES input.file;1  input.file;2 /CHANGE_BAR=#
```

The following placement displays *input.file;1* with the pound sign (#) preceding any lines that differ from *input.file;2*:

```
$ DIFFERENCES input.file;1/CHANGE_BAR=#  input.file;2
```

The following placement displays *input.file;1* with the percent sign (%) preceding any lines that differ from *input.file;2*, and also displays *input.file;2* with the pound sign (#) preceding any lines that differ from *input.file;1*:

```
$ DIFFERENCES input.file;1/CHANGE_BAR=% input.file;2/CHANGE_BAR=#
```

- If you do not specify a change bar character, the default is an exclamation point (!) for ASCII output.
- If you specify hexadecimal or octal output (see the description of the /MODE qualifier), the change bar character is ignored and differences are marked by a ‘‘***CHANGE***’’ string in the record header. The keyword NONUMBER suppresses line numbers in the listing.
- If neither the NUMBER nor the NONUMBER keyword is specified, the default is controlled by the /[NO]NUMBER command qualifier.
- If you specify only one option, you can omit the parentheses.
- If you use an exclamation point (!) as the specified character, you must enclose it in quotation marks (‘‘ ’’); for example, /CHANGE_BAR=(‘‘!’’,NUMBER).

/COMMENT_DELIMITER[=(character[,...])]

Ignores lines starting with a specified comment character.

If the comment character is an exclamation point or semicolon (;), it can appear anywhere in the line and characters to the right of the character are ignored. If you specify just one character, you can omit the parentheses. Lowercase characters are automatically converted to uppercase unless they are enclosed in quotation marks. Nonalphanumeric characters (such as ! and ,) must be enclosed in quotation marks. You can specify up to 32 comment characters by typing the character itself or one of the following keywords. (Keywords can be abbreviated provided that the resultant keyword is not ambiguous and has at least 2 characters; single letters are treated as delimiters.)

Keyword	Character
COLON	Colon (:)
COMMA	Comma (,)
EXCLAMATION	Exclamation point (!)
FORM_FEED	Form feed
LEFT	Left bracket ([)
RIGHT	Right bracket (])
SEMI_COLON	Semicolon (;)
SLASH	Slash (/)
SPACE	Space
TAB	Tab

The /COMMENT_DELIMITER qualifier is used with or without the /IGNORE=COMMENTS qualifier to indicate which comments are to be ignored.

If both the uppercase and lowercase forms of a letter are to be used as comment characters, the letter must be specified twice, once in uppercase and once in lowercase. If you do not include either a comment character or a keyword with the /COMMENT_DELIMITER qualifier, the DIFFERENCES command assumes a default comment character based on the file type. For some file types (.COB and .FOR), the default comment characters are considered valid delimiters only if they appear in the first column of a line. Multicharacter comment characters are not allowed.

The following characters are the default comment delimiters for files with the specified file types:

File Type	Default Comment Character
.B2S, .B32, .BAS, .BLI	!
.CBL, .CMD	! and ;
.COB	* or / in the first column
.COM, .COR	!
.FOR	! anywhere and C, D, c, d in the first column
.HLP	!
.MAC, .MAR	;
.R32, .REQ	!

/EXACT

Use with the /PAGE=SAVE and /SEARCH qualifiers to specify a search string that must match the search string exactly and must be enclosed with quotation marks (‘‘ ’’).

If you specify the /EXACT qualifier without the /SEARCH qualifier, exact search mode is enabled when you set the search string with the Find (E1) key.

/HIGHLIGHT[=keyword]
/NOHIGHLIGHT (default)

Use with the /PAGE=SAVE and /SEARCH qualifiers to specify the type of highlighting you want when a search string is found. When a string is found, the entire line is highlighted. You can use the following keywords: BOLD, BLINK, REVERSE, and UNDERLINE. BOLD is the default highlighting.

/IGNORE=(keyword[,...])

Inhibits the comparison of the specified characters, strings, or records; also controls whether the comparison records are output to the listing file as edited records or exactly as they appeared in the input file. If you specify only one keyword, you can omit the parentheses. The keyword parameter refers to either a character or a keyword. The first set of keywords determines what, if anything, is ignored during file comparison; the second set of keywords determines whether or not ignored characters are included in the output. The following keywords are valid options for the /IGNORE qualifier:

Keyword	Item Ignored
BLANK_LINES	Blank lines between data lines.
CASE	Case of the text being compared.

Keyword	Item Ignored
COMMENTS	Data following a comment character. (Use the /COMMENT_DELIMITER qualifier to designate one or more nondefault comment delimiters.)
FORM_FEEDS	Form feed character.
HEADER[=n]	First *n* records of the file, beginning with a record whose first character is a form feed. The first record is not ignored if the only character it contains is a form feed. (*N* indicates the number of records and defaults to 2. A record with a single form feed is not counted.)
SPACING	Extra blank spaces or tabs within data lines.
TRAILING_SPACES	Space and tab characters at the end of a data line.

Keyword	Status of Ignored Items in Output
EDITED	Omits ignored characters from the output records.
EXACT	Includes ignored characters in the output records.
PRETTY	Formats output records.

Each data line is checked for COMMENTS, FORM_FEEDS, HEADER, and SPACING before it is tested for TRAILING_SPACES and then BLANK_LINES. Therefore, if you direct the DIFFERENCES command to ignore COMMENTS, TRAILING_SPACES, and BLANK_LINES, it ignores a record that contains several spaces or blank lines followed by a comment.

By default, the DIFFERENCES command compares every character in each file and reports all differences. Also, by default, the DIFFERENCES command lists records in the output file with all ignored characters deleted.

If you specify the /PARALLEL qualifier, output records are always formatted. To format output records, specify the following characters:

Character	Formatted Output
Tab (Ctrl/I)	1–8 spaces
Return (Ctrl/M)	<CR>
Line feed (Ctrl/J)	<LF>
Vertical tab (Ctrl/K)	<VT>
Form feed (Ctrl/L)	<FF>
Other nonprinting characters	. (period)

/MATCH=size

Specifies the number of records that should indicate matching data after a difference is found. By default, after the DIFFERENCES command finds unmatched records, it assumes that the files once again match after it finds three sequential records that match. Use the /MATCH qualifier to override the default match size of 3.

You can increase the /MATCH qualifier value if you feel that the DIFFERENCES command is incorrectly matching sections of the master and revision input files after it has detected a difference.

/MAXIMUM_DIFFERENCES=n
Terminates the DIFFERENCES command after the specified number of unmatched records (specified with the *n* parameter) is found.

The number of unmatched records is determined by finding the maximum number of difference records for each difference section and adding them together.

If the DIFFERENCES command reaches the maximum number of differences that you specify, it will output only those records that were detected before the maximum was reached. Also, it will output, at most, one listing format and return a warning message.

By default, there is no maximum number of differences. All records in the specified input files are compared.

/MERGED[=n]
Specifies that the output file contain a merged list of differences with the specified number of matched records listed after each group of unmatched records. The value of the parameter *n* must be less than or equal to the number specified in the /MATCH qualifier. By default, the DIFFERENCES command produces a merged listing with one matched record listed after each set of unmatched records (that is, /MERGED=1). If the /MERGED, the /SEPARATED, or the /PARALLEL qualifier is not specified, the resulting output is merged, with one matched record following each unmatched record.

Use the /MERGED qualifier to override the default value of the parameter *n*, or to include a merged listing with other types of output.

/MODE=(radix[,...])
Specifies the format of the output. You can request that the output be formatted in one or more radix modes by specifying the following keywords, which may be abbreviated: ASCII (default), HEXADECIMAL, or OCTAL. If you specify only one radix, you can omit the parentheses.

By default, the DIFFERENCES command writes the output file in ASCII. If you specify more than one radix, the output listing contains the file comparison in each specified radix. When you specify two or more radix modes, separate them with commas.

If you specify the /PARALLEL or the /SLP qualifier, the /MODE qualifier is ignored for that listing form.

/NUMBER (default)
/NONUMBER
Includes line numbers in the listing of DIFFERENCES.

/OUTPUT[=filespec]
Specifies an output file to receive the list of differences. By default, the output is written to the current SYS$OUTPUT device. If the *filespec* parameter is not specified, the output is directed to the first input file with a file type .DIF. The asterisk (*) and the percent sign (%) wildcard characters are not allowed.

When you specify the /OUTPUT qualifier, you can control the defaults applied to the output file specification as described in the *OpenVMS User's Manual*. The default output file type is .DIF.

/PAGE[=keyword]
/NOPAGE (default)
Controls the display of difference information on the screen.

You can use the following keywords with the /PAGE qualifier:

CLEAR_SCREEN	Clears the screen before each page is displayed.
SCROLL	Displays information one line at a time.
SAVE[=*n*]	Enables screen navigation of information, where *n* is the number of pages to store.

The /PAGE=SAVE qualifier allows you to navigate through screens of information. The /PAGE=SAVE qualifier stores up to 5 screens of up to 255 columns of information. When you use the /PAGE=SAVE qualifier, you can use the following keys to navigate through the information:

Key Sequence	Description
Up arrow (↑), Ctrl/B	Scroll up one line.
Down arrow (↓)	Scroll down one line.
Left arrow (←)	Scroll left one column.
Right arrow (→)	Scroll right one column.
Find (E1)	Specify a string to find when the information is displayed.
Insert Here (E2)	Scroll right one half screen.
Remove (E3)	Scroll left one half screen.
Select (E4)	Toggle 80/132 column mode.
Prev Screen (E5)	Get the previous page of information.
Next Screen (E6), Return, Enter, Space	Get the next page of information.
F10, Ctrl/Z	Exit. (Some utilities define these differently.)
Help (F15)	Display utility help text.
Do (F16)	Toggle the display to oldest/newest page.
Ctrl/W	Refresh the display.

The /PAGE qualifier is not compatible with the /OUTPUT qualifier.

/PARALLEL[=n]

Lists the records with differences side by side. The value of the parameter *n* specifies the number of matched records to merge after each unmatched record; it must be a non-negative decimal number less than or equal to the number specified in the /MATCH qualifier.

By default, the DIFFERENCES command does not list records after each list of unmatched records. Also by default, the DIFFERENCES command creates only a list of merged differences.

/SEARCH="string"

Use with the /PAGE=SAVE qualifier to specify a string that you want to find in the information being displayed. Quotation marks are required for the /SEARCH qualifier, if you include spaces in the text string.

You can also dynamically change the search string by pressing the Find key (E1) while the information is being displayed. Quotation marks are not required for a dynamic search.

/SEPARATED[=MASTER, REVISION]
Lists sequentially only the records from the specified file that contain differences. Use the MASTER keyword to list the differences in the first input file specified; use the REVISION keyword to list the difference in the second input file specified.

By default, the DIFFERENCES command creates only a merged list of differences.

/SLP
Requests that the DIFFERENCES command produce an output file suitable for input to the SLP editor. If you use the /SLP qualifier, you cannot specify any of the following output file qualifiers: /MERGED, /PARALLEL, /SEPARATED, or /CHANGE_BAR.

Use the output file produced by the SLP qualifier as input to SLP to update the master input file, that is, to make the master input file match the revision input file.

When you specify the /SLP qualifier and you do not specify the /OUTPUT qualifier, the DIFFERENCES command writes the output file to a file with the same file name as the master input file with the file type DIF.

/WIDTH=n
Specifies the width of the lines in the output file. The default is 132 characters. If output is written to the terminal, the /WIDTH qualifier is ignored and the terminal line width is used.

Use the SET TERMINAL command to change the terminal line width.

/WINDOW=size
Searches the number of records specified by the *size* parameter, before a record is declared as unmatched. By default, the DIFFERENCES command searches to the ends of both input files before listing a record as unmatched.

The window size is the minimum size of a differences section that will cause the DIFFERENCES command to lose synchronization between the two input files.

/WRAP
/NOWRAP (default)
Use with the /PAGE=SAVE qualifier to limit the number of columns to the width of the screen and to wrap lines that extend beyond the width of the screen to the next line.

The /NOWRAP qualifier extends lines beyond the width of the screen and can be seen when you use the scrolling (left and right) features provided by the /PAGE=SAVE qualifier.

Examples

1.
```
$ DIFFERENCES EXAMPLE.TXT
************
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
    1   DEMONSTRATION
    2   OF V3.0 DIFFERENCES
    3   UTILITY
******
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
    1   DEMONSTRETION
    2   OF VMS DIFFERENCES
    3   UTILITY
************
Number of difference sections found: 1
Number of difference records found: 2
DIFFERENCES/MERGED=1-
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
```

In this example, the DIFFERENCES command compares the contents of the two most recent versions of the file EXAMPLE.TXT in the current default directory. The DIFFERENCES command compares every character in every record and displays the results at the terminal.

2.
```
$ DIFFERENCES/PARALLEL/WIDTH=80/COMMENT_DELIMITER="V" EXAMPLE.TXT
--------------------------------------------------------------------------------
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2  |  File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
------------------ 1 ------------------------------------ 1 -----------------
DEMONSTRATION                          |  DEMONSTRETION
--------------------------------------------------------------------------------
Number of difference sections found: 1
Number of difference records found: 1
DIFFERENCES/IGNORE=(COMMENTS)/COMMENT_DELIMITER=("V")/WIDTH=80/PARALLEL-
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2-
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
```

The DIFFERENCES command compares the same files as in Example 1, but ignores all comments following the first ‘‘V’’ encountered by the DIFFERENCES command. The command also specifies that an 80-column parallel list of differences be displayed.

3.
```
$ DIFFERENCES/WIDTH=80/MODE=(HEX,ASCII) EXAMPLE.TXT/CHANGE_BAR
************
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
    1 ! DEMONSTRATION
    2 ! OF V3.0 DIFFERENCES
    3   UTILITY
************
************
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
RECORD NUMBER 1 (00000001) LENGTH 14 (0000000E)  ***CHANGE***
     204E 4F495441 5254534E 4F4D4544 DEMONSTRATION .. 000000
RECORD NUMBER 2 (00000002) LENGTH 19 (00000013)  ***CHANGE***
 4E455245 46464944 20302E33 5620464F OF V3.0 DIFFEREN 000000
                               534543 CES............. 000010
RECORD NUMBER 3 (00000003) LENGTH 7 (00000007)
                     595449 4C495455 UTILITY......... 000000
************
Number of difference sections found: 1
Number of difference records found: 2
DIFFERENCES /WIDTH=80/MODE=(HEX,ASCII)
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2/CHANGE_BAR-
       DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
```

The DIFFERENCES command compares the same files as in Example 1, but lists the differences in both hexadecimal and ASCII formats. The command also specifies that default change bars be used in the output. The default change bar notation for the hexadecimal output is ***CHANGE***. For the ASCII output, the default change bar character is the exclamation point.

4.
```
$ DIFFERENCES/OUTPUT BOSTON::DISK2:TEST.DAT OMAHA::DISK1:[PGM]TEST.DAT
```

The DIFFERENCES command compares two remote files and displays any differences found. The first file is TEST.DAT on remote node BOSTON. The second file is also named TEST.DAT on remote node OMAHA. The DIFFERENCES output is located in the file DISK1:[PGM]TEST.DIF.

DIRECTORY

Provides a list of files or information about a file or group of files.

Requires execute (E) access to look up files you know the names of, read (R) access to read or list a file or to use a file name with the asterisk (*) and the percent sign (%) wildcard characters to look up files.

Format

DIRECTORY [filespec[,...]]

DIRECTORY/FTP directorySpec

Parameter

filespec[,...]

Specifies one or more files to be listed. The syntax of a file specification determines which files will be listed, as follows:

- If you do not enter a file specification, the DIRECTORY command lists all versions of the files in the current default directory.
- If you specify only a device name, the DIRECTORY command uses your default directory specification.
- Whenever the file specification does not include a file name, a file type, and a version number, all versions of all files in the specified directory are listed.
- If a file specification contains a file name or a file type, or both, and no version number, the DIRECTORY command lists all versions.
- If a file specification contains only a file name, the DIRECTORY command lists all files in the current default directory with that file name, regardless of file type and version number.
- If a file specification contains only a file type, the DIRECTORY command lists all files in the current default directory with that file type, regardless of file name and version number.

The asterisk (*) and the percent sign (%) wildcard characters can be used in the directory specification, file name, file type, or version number fields of a file specification to list all files that satisfy the components you specify. If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+).

directorySpec

Specifies the standard DECnet remote file specification. Use a quoted file string to preserve the case (for case-sensitive systems such as UNIX) and to identify a foreign device/directory specification. See the /FTP qualifier for more information.

Description

The DIRECTORY command lists the files contained in a directory. When you use certain qualifiers with the command, additional information is displayed, along with the names of the files.

The output of the DIRECTORY command depends on certain formatting qualifiers and their defaults. These qualifiers are as follows: /COLUMNS, /DATE, /FULL, /OWNER, /PROTECTION, and /SIZE. However, the files are always listed in alphabetical order, with the highest numbered versions listed first.

In studying the qualifiers and the capabilities they offer, watch for qualifiers that work together and for qualifiers that override other qualifiers. For example, if you specify the /FULL qualifier, the system cannot display all the information in more than one column. Thus, if you specify both the /COLUMNS and /FULL qualifiers, the number of columns you request is ignored.

You can also select other languages and formats that have been defined on your systems with international date and time formatting routines available in the run-time library. See the *OpenVMS RTL Library (LIB$) Manual*.

Qualifiers

/ACL

Controls whether the access control list (ACL) is displayed for each file. By default, the DIRECTORY command does not display the ACL for each file. Access control entries (ACEs) that were created with the hidden option are displayed only if the SECURITY privilege is turned on. The /ACL qualifier overrides the /COLUMNS qualifier.

For further information, see the *OpenVMS Guide to System Security*.

/BACKUP

Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /BACKUP qualifier selects files according to the dates of their most recent backups. This qualifier is incompatible with the /CREATED, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /BEFORE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/BRIEF (default)

Displays only a file's name, type, and version number. The brief format lists the files in alphabetical order from left to right on each line, in descending version number order. You can use the /ACL, /DATE, /FILE_ID, /FULL, /NOHEADING, /OWNER, /PROTECTION, /SECURITY, and /SIZE qualifiers to expand a brief display.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

Specify the UIC by using standard UIC format as described in the *OpenVMS User's Manual*.

For further information, see the *OpenVMS Guide to System Security*.

/COLUMNS=n
Specifies the number of columns in a brief display. The default is four. However, you can request as many columns as you like, restricted by the value of the /WIDTH qualifier. The /COLUMNS qualifier is incompatible with the /ACL, /FULL, and /SECURITY qualifiers.

The number of columns actually displayed depends on the amount of information requested for each column and the display value of the /WIDTH qualifier. The system displays only as many columns as can fit within the default or specified display width, regardless of how many columns you specify with the /COLUMNS qualifier.

The DIRECTORY command truncates long file names only when you specify more than one column and you have asked for additional information to be included in each column. The default file name size is 19 characters. Use the /WIDTH qualifier to change the default. When a file name is truncated, the system displays one less character than the file name field size and inserts a vertical bar in the last position. For example, if the file name is SHOW_QUEUE_CHARACTERISTICS, and if you requested DIRECTORY to display both file name and size in each column, the display for that file would be SHOW_QUEUE_CHARACT| 120.

/CREATED (default)
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the /BACKUP, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/DATE[=option]
/NODATE (default)
Includes the creation, last modification, expiration, backup, effective, or recording date for each specified file; the default is the /NODATE qualifier. If you use the /DATE qualifier without an option, the creation date is provided. Possible options are as follows:

Option	Description
ALL	Specifies all optional dates in the following order: creation, last modification, expiration, backup, effective, and recording.
BACKUP	Specifies the last backup date.
CREATED	Specifies the creation date.
EFFECTIVE	Specifies the effective date the contents are valid (ISO 9660).
EXPIRED	Specifies the expiration date.
MODIFIED	Specifies the last modification date.
RECORDING	Specifies the recording date on the media (ISO 9660).

/EXACT
Use with the /PAGE=SAVE and /SEARCH qualifiers to specify a search string that must match the search string exactly and must be enclosed with quotation marks (‘‘ ’’).

If you specify the /EXACT qualifier without the /SEARCH qualifier, exact search mode is enabled when you set the search string with the Find (E1) key.

/EXCLUDE=(filespec[,...])
Excludes the specified files from the DIRECTORY command. You can include a directory but not a device in the file specification. The asterisk (*) and the percent sign (%) wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /EXPIRED qualifier selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the /BACKUP, /CREATED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time qualifiers, the default is the /CREATED qualifier.

/FILE_ID
Controls whether the file identification (FID) number is displayed. By default, the FID is not displayed unless the /FULL qualifier is specified.

/FTP
Invokes the directory (*dir* or *ls*) operation of the FTP utility. The DIRECTORY /FTP command writes a listing of the contents of the specified remote directory to the local host over a TCP/IP connection by invoking the FTP utility.

/FULL
Displays the following information for each file:

File name
File type
Version number
File identification number (FID)
Number of blocks used
Number of blocks allocated
File owner's user identification code (UIC)
Date of creation
Date last modified and revision number
Date of expiration
Date of last backup
Date of effective usage
Date of recording on media
File organization
Shelved state
File attributes
Record format
Record attributes
RMS attributes
Journaling information
File protection
Access control list (ACL)
Value of the stored semantics tag (where applicable)

/GRAND_TOTAL
Displays only the totals for all files and directories that have been specified. Suppresses both the per-directory total and individual file information. (See the /TRAILING qualifier for information on displaying directory totals.)

/HEADING
/NOHEADING
Controls whether heading lines consisting of a device description and directory specification are printed. The default output format provides this heading. When the /NOHEADING qualifier is specified, the display is in single-column format and the device and directory information appears with each file name. The /NOHEADING qualifier overrides the /COLUMNS qualifier.

The combination of the /NOHEADING and /NOTRAILING qualifiers is useful in command procedures where you want to create a list of complete file specifications for later operations.

/HIGHLIGHT[=keyword]
/NOHIGHLIGHT (default)
Use with the /PAGE=SAVE and /SEARCH qualifiers to specify the type of highlighting you want when a search string is found. When a string is found, the entire line is highlighted. You can use the following keywords: BOLD, BLINK, REVERSE, and UNDERLINE. BOLD is the default highlighting.

/MODIFIED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /MODIFIED qualifier selects files according to the dates on which they were last modified. This qualifier is incompatible with the /BACKUP, /CREATED, and /EXPIRED qualifiers, which also allow you to select files according to time attributes. If you specify none of these four time modifiers, the default is the /CREATED qualifier.

/OUTPUT[=filespec]
/NOOUTPUT
Controls where the output of the command is sent. By default, the display is written to the current SYS$OUTPUT device. The asterisk (*) and the percent sign (%) wildcard characters are not allowed.

If you enter the /OUTPUT qualifier with a partial file specification (for example, /OUTPUT=[JONES]), DIRECTORY is the default file name and LIS the default file type. If you enter the /NOOUTPUT qualifier, output is suppressed.

If the output will be written to a file in the same directory, the output file name will appear in the directory listing.

/OWNER
/NOOWNER (default)
Controls whether the file owner's user identification code (UIC) is listed.

The default size of the owner field is 20 characters. If the file owner's UIC exceeds the length of the owner field, the information will be truncated. The size of this field can be altered by specifying /WIDTH=OWNER, along with a value for the owner field. For more information, see the description of the /WIDTH qualifier.

/PAGE[=keyword]
/NOPAGE (default)
Controls the display of directory information on the screen.

You can use the following keywords with the /PAGE qualifier:

CLEAR_SCREEN	Clears the screen before each page is displayed.
SCROLL	Displays information one line at a time.
SAVE[=*n*]	Enables screen navigation of information, where *n* is the number of pages to store.

The /PAGE=SAVE qualifier allows you to navigate through screens of information. The /PAGE=SAVE qualifier stores up to 5 screens of up to 255 columns of information. When you use the /PAGE=SAVE qualifier, you can use the following keys to navigate through the information:

Key Sequence	Description
Up arrow (↑), Ctrl/B	Scroll up one line.
Down arrow (↓)	Scroll down one line.
Left arrow (←)	Scroll left one column.
Right arrow (→)	Scroll right one column.
Find (E1)	Specify a string to find when the information is displayed.
Insert Here (E2)	Scroll right one half screen.
Remove (E3)	Scroll left one half screen.
Select (E4)	Toggle 80/132 column mode.
Prev Screen (E5)	Get the previous page of information.
Next Screen (E6), Return, Enter, Space	Get the next page of information.
F10, Ctrl/Z	Exit. (Some utilities define these differently.)
Help (F15)	Display utility help text.
Do (F16)	Toggle the display to oldest/newest page.
Ctrl/W	Refresh the display.

The /PAGE qualifier is not compatible with the /OUTPUT qualifier.

/PRINTER
Puts the display in a file and queues the file to SYS$PRINT for printing under the name given by the /OUTPUT qualifier. If you do not specify the /OUTPUT qualifier, output is directed to a temporary file named DIRECTORY.LIS, which is queued for printing and then is deleted.

/PROTECTION
/NOPROTECTION (default)
Controls whether the file protection for each file is listed.

/SEARCH="string"

Use with the /PAGE=SAVE qualifier to specify a string that you want to find in the information being displayed. Quotation marks are required for the /SEARCH qualifier, if you include spaces in the text string.

You can also dynamically change the search string by pressing the Find key (E1) while the information is being displayed. Quotation marks are not required for a dynamic search.

/SECURITY

Controls whether information about file security is displayed; using the /SECURITY qualifier is equivalent to using the /ACL, /OWNER, and /PROTECTION qualifiers together. ACEs that were created with the hidden option are displayed only if the SECURITY privilege is turned on.

For further information, see the *OpenVMS Guide to System Security*.

/SELECT=(keyword[,...])

Allows you to select files for display. Choose one of the following keywords:

ACL NOACL	Displays files that have an associated ACL or files that do not (NOACL keyword).
FILE=(option[,...])	Displays portions of the file specification. The /SELECT=FILE qualifier is used to turn off specific portions by explicit or implicit specification of the options. Possible options are: [NO]NODE [NO]DEVICE [NO]DIRECTORY [NO]NAME [NO]TYPE [NO]VERSION /SELECT=FILE qualifier cannot be used with the /FULL qualifier.
ONLINE NOONLINE	Displays files that are online or shelved.
SHELVABLE NOSHELVABLE	Displays files that are shelvable or not shelvable.

SIZE=(option[,...])

Displays files according to their size. Possible options are:

Option	Description
MAXIMUM=*n*	Displays files that have fewer blocks than the value of *n*, which defaults to 1,073,741,823. Use with MINIMUM=n to specify a size range for files to be displayed.
MINIMUM=*n*	Displays files that have blocks equal to or greater than the value of *n*. Use with MAXIMUM=n to specify a size range for files to be displayed.
(MINIMUM=*n*, MAXIMUM=*n*)	Displays files whose block size falls within the specified MINIMUM and MAXIMUM range.

By default, file selection is based on other criteria.

/SHELVED_STATE
Displays whether the file is shelved or online.

/SINCE[=time]
Selects only those files dated after the specified time. You can specify time as an absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /SINCE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information on specifying time values, see the *OpenVMS User's Manual*.

/SIZE[=option]
/NOSIZE (default)
Displays the size in blocks of each file. If you omit the option parameter, the default lists the file size in blocks used (USED). Specify one of the following options:

ALL	Lists the file size both in blocks allocated and blocks used.
ALLOCATION	Lists the file size in blocks allocated.
USED	Lists the file size in blocks used.

The size of this field can be altered by supplying the size value of the /WIDTH qualifier.

/TIME[=option]
/NOTIME (default)
The same as the /DATE qualifier: includes the backup, creation, expiration, or modification time for each specified file; the default is the /NOTIME qualifier. If you use the /TIME qualifier without an option, the creation time is provided. Possible options are as follows:

Option	Description
ALL	Specifies creation, expiration, backup, and last modification times.
BACKUP	Specifies the last backup time.
CREATED	Specifies the creation time.
EFFECTIVE	Specifies the effective time the contents are valid.
EXPIRED	Specifies the expiration time.
MODIFIED	Specifies the last modification time.
RECORDING	Specifies the recording time on the media.

/TOTAL
Displays only the directory name and total number of files.

By default, the output format is determined by the /BRIEF qualifier, which gives this total but also lists all the file names, file types, and their version numbers.

/TRAILING
/NOTRAILING
Controls whether trailing lines that provide the following summary information are displayed:

- Number of files listed
- Total number of blocks used per directory
- Total number of blocks allocated
- Total number of directories and total blocks used or allocated in all directories (only if more than one directory is listed)

By default, the output format includes most of this summary information. The /SIZE and /FULL qualifiers determine more precisely what summary information is included.

When used alone, the /TRAILING qualifier lists the number of files in the directory. When used with the /SIZE qualifier, the /TRAILING qualifier lists the number of files and the number of blocks (displayed according to the option of the /SIZE qualifier, FULL or ALLOCATION). When used with the /FULL qualifier, the /TRAILING qualifier lists the number of files as well as the number of blocks used and allocated. If more than one directory is listed, the summary includes the total number of directories, the total number of blocks used, and the total number of blocks allocated.

/VERSIONS=n
Specifies the number of versions of a file to be listed. The default is all versions of each file. A value less than 1 is not allowed.

/WIDTH=(keyword[,...])
Formats the width of the display. If you specify only one keyword, you can omit the parentheses. Possible keywords are as follows:

DISPLAY=*n* Specifies the total width of the display as an integer in the range 1 to 256 and defaults to zero (setting the display width to the terminal width). If the total width of the display exceeds the terminal width, the information will be truncated.

FILENAME=*n* Specifies the width of the file name field; defaults to 19 characters. If you request another piece of information to be displayed along with the file name in each column, file names that exceed the *n* parameter cause the line to wrap after the file name field. (See the /COLUMNS qualifier.)

OWNER=*n* Specifies the width of the owner field; defaults to 20 characters. If the owner's user identification code (UIC) exceeds the length of the owner field, the information will be truncated.

SIZE=*n* Specifies the width of the size field; defaults to 6 characters on systems prior to OpenVMS Version 6.0; the default is 7 characters on OpenVMS Version 6.0 systems or higher. If the file size exceeds the length of the size field, the field is filled with asterisks.

/WRAP
/NOWRAP (default)
Use with the /PAGE=SAVE qualifier to limit the number of columns to the width of the screen and to wrap lines that extend beyond the width of the screen to the next line.

The /NOWRAP qualifier extends lines beyond the width of the screen and can be seen when you use the scrolling (left and right) features provided by the /PAGE=SAVE qualifier.

Examples

1.
```
$ DIRECTORY AVERAGE.*

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6      AVERAGE.FOR;6      AVERAGE.LIS;4     AVERAGE.OBJ;12

Total of 4 files.
```

   In this example, the DIRECTORY command lists all files with the file name AVERAGE and any file type.

2. `$ DIRECTORY/SIZE=USED/DATE=CREATED/VERSIONS=1/PROTECTION  AVERAGE`

```
Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6       6        19-APR-1995 15:43:02.10 (RE,RE,RWED,RE)
AVERAGE.FOR;6       2        19-APR-1995 10:29:53.37 (RE,RE,RWED,RE)
AVERAGE.LIS;4       5        19-APR-1995 16:27:27.19 (RE,RE,RWED,RE)
AVERAGE.OBJ;6       2        19-APR-1995 16:27:44.23 (RE,RE,RWED,RE)

Total of 4 files, 15 blocks.
```

In this example, the DIRECTORY command lists the number of blocks used, the creation date, and the file protection code for the highest version number of all files named AVERAGE in the current directory.

3. `$ DIRECTORY/FULL DISK$GRIPS_2:[VMS.TV]DEMO.EXE`

```
Directory DISK$GRIPS_2:[VMS.TV]

DEMO.EXE;1                      File ID:  (36,11,0)
Size:           390/390         Owner:    [0,0]
Created:  12-NOV-1995 11:45:19.00
Revised:  14-DEC-1995 15:45:19.00 (34)
Expires:   <None specified>
Backup:   28-NOV-1995 04:00:12.22
Effective: <None specified>
Recording: <None specified>
File organization:  Sequential
Shelved state:      Online
File attributes:    Allocation: 390, Extend: 0, Global buffer count: 0,
                    Version limit: 0, Backups disabled, Not shelvable
Record format:      Fixed length 512 byte records
Record attributes:  None
RMS attributes:     None
Journaling enabled: None
File protection:    System:RE, Owner:RE, Group:RE, World:RE
Access Cntrl List:  None

Total of 1 file, 390/390 blocks.
```

The example illustrates the DIRECTORY/FULL command.

4. `$ DIRECTORY/VERSIONS=1/COLUMNS=1 AVERAGE.*`

The DIRECTORY command in this example lists only the highest version of each file named AVERAGE in the current default directory. The format is brief and restricted to one column. Heading and trailing lines are provided.

5. `$ DIRECTORY BLOCK%%%`

The DIRECTORY command in this example locates all versions and types of files in the default device and directory whose names begin with the letters BLOCK and end with any three additional characters. The default output format is brief, four columns, with heading and trailing lines.

6. `$ DIRECTORY/EXCLUDE=(AVER.DAT;*,AVER.EXE;*) [*...]AVER`

The DIRECTORY command in this example lists and totals all versions and types of files named AVER in all directories and subdirectories on the default disk, except any files named AVER.DAT and AVER.EXE.

7.
```
$ DIRECTORY/SIZE=ALL FRESNO::DISK1:[TAYLOR]*.COM
```

   The DIRECTORY command in this example lists all versions of all files with the file type COM in the directory TAYLOR on node FRESNO and device DISK1. The listing includes the file size both in blocks used and in blocks allocated for each file.

8.
```
$ DIRECTORY-
_$ /MODIFIED/SINCE=14-DEC-1995:01:30/SIZE=ALL/OWNER-
_$ /PROTECTION/OUTPUT=UPDATE/PRINTER [A*]
```

   The DIRECTORY command in this example locates all files that have been modified since 1:30 a.m. on December 14, 1995, and that reside on the default disk in all directories whose names begin with the letter A. It formats the output to include all versions, the size used and size allocated, the date last modified, the owner, and the protection codes. The output is directed to a file named UPDATE.LIS, which is queued automatically to the default printer queue and then is deleted.

9.
```
$ DIRECTORY/SHELVED_STATE

Directory MYDISK:[SMITH]

MYFILE.TXT;2            Online
NOT_SHELVED.TXT;1       Online
SHELVED.TXT             Shelved

Total of 3 files.
```

   The DIRECTORY command in this example lists all the files in a directory and shows whether a file is shelved, online, or remote.

DISABLE AUTOSTART

Disables the autostart feature on a node for all autostart queues managed by the specific queue manager. By default, this command uses the /QUEUES qualifier.

Requires OPER (operator) privileges.

For more information on autostart queues, see the chapter on batch and print queues in the *OpenVMS System Manager's Manual*.

Format

DISABLE AUTOSTART[/QUEUES]

Parameters

None.

Description

The DISABLE AUTOSTART/QUEUES command notifies the queue manager to perform the following tasks on the affected node:

- Mark all of the queue manager's autostart queues as ''stop pending'' in preparation for a planned shutdown.
- Prevent any of the queue manager's autostart queues from failing over to the node.
- Upon completion of any jobs currently executing on any of that queue manager's autostart queues, force the queue to fail over to the next available node in the queue's failover list (if any) on which autostart is enabled. (For information on failover lists for autostart queues, see the /AUTOSTART_ON qualifier for the INITIALIZE/QUEUE command.)

Autostart queues on the node that do not have a failover list, or for which no failover node is enabled for autostart, are stopped upon completion of any current jobs. These stopped queues remain activated for autostart. The queue manager will restart these stopped autostart queues when the ENABLE AUTOSTART command is entered for the affected node or a node to which the queue can fail over.

By default the command affects the node on which it is entered. Specify the /ON_NODE qualifier to disable autostart on a different node.

The DISABLE AUTOSTART/QUEUES command is included in the node shutdown command procedure SHUTDOWN.COM. If you shutdown a node without using SHUTDOWN.COM, and the node is running autostart queues, you might want to enter the DISABLE AUTOSTART/QUEUES command first.

The DISABLE AUTOSTART/QUEUES command only affects autostart queues.

Qualifiers

/NAME_OF_MANAGER=name
Specifies the name of the queue manager controlling the autostart queues you want to disable. The qualifier allows the autostart feature to be used differently for different sets of queues.

If the /NAME_OF_MANAGER qualifier is omitted, the default queue manager name SYS$QUEUE_MANAGER is used. For more information on multiple queue managers, see the *OpenVMS System Manager's Manual*.

/ON_NODE=nodename
Specifies a node in a VMScluster system. Use this qualifier to disable autostart on a node other than the one from which you enter the command.

/QUEUES
Specifies that autostart is to be disabled for queues. (This qualifier is used by default.)

Examples

1.
```
$ INITIALIZE/QUEUE/BATCH/START/AUTOSTART_ON=SATURN:: BATCH_1
$ ENABLE AUTOSTART/QUEUES
   .
   .
   .
$ DISABLE AUTOSTART/QUEUES
```

In this example, the INITIALIZE/QUEUE command creates an autostart queue BATCH_1, capable of executing on node SATURN. The /START qualifier activates the queue for autostart. The ENABLE/AUTOSTART /QUEUES command (executed on node SATURN) enables autostart on the node, causing the queue (and any other active autostart queues on the node)to begin executing jobs.

The DISABLE AUTOSTART command (executed on node SATURN) stops autostart queues on the node, and prevents any queues from failing over to the node.

This command only affects queues managed by the default queue manager SYS$QUEUE_MANAGER because the /NAME_OF_MANAGER qualifier is not specified.

Because BATCH_1 is set up to run only on one node, the queue cannot fail over to another node and therefore is stopped. However, the queue remains active for autostart and will be started when the ENABLE AUTOSTART command is entered for node SATURN. No START/QUEUE command is needed to restart BATCH_1 unless autostart of the queue is deactivated with the STOP/QUEUE/NEXT or STOP/QUEUE/RESET command.

2.
```
$DISABLE AUTOSTART/QUEUES/ON_NODE=JADE
```

The DISABLE AUTOSTART/QUEUES command in this example disables autostart on the VMScluster node JADE. This command can be entered from any node in the cluster.

DISCONNECT

Breaks the connection between a physical terminal and a virtual terminal. After the physical terminal is disconnected, both the virtual terminal and the process using it remain on the system.

Requires that your physical terminal is connected to a virtual terminal.

Format

DISCONNECT

Parameters

None.

Description

Use the DISCONNECT command to disconnect a physical terminal from a virtual terminal and its associated process. The virtual terminal and the process remain on the system, so you can use the CONNECT command to reconnect to the process later. (For more information about virtual terminals and how to connect to them, see the description of the CONNECT command.) To terminate a process connected to a virtual terminal, use the LOGOUT command.

After you are disconnected from a virtual terminal, you can use the physical terminal to log in again.

You can use the DISCONNECT command only if your physical terminal is connected to a virtual terminal.

Qualifier

/CONTINUE
/NOCONTINUE (default)
Controls whether the CONTINUE command is executed in the current process just before connecting to another process. This procedure permits an interrupted image to continue processing after the disconnection until the process needs terminal input or attempts to write to the terminal. At that point, the process waits until the physical terminal is reconnected to the virtual terminal.

Examples

1.
```
$ DISCONNECT
```

   This command disconnects a physical terminal from a virtual terminal, but does not log the process out. Now you can use the physical terminal to log in again.

2.
```
$ RUN PAYROLL
```
   Ctrl/Y
```
$ DISCONNECT/CONTINUE
```

   In this example, the RUN command is entered from a physical terminal that is connected to a virtual terminal. After the image PAYROLL.EXE is interrupted, the DISCONNECT command disconnects the physical and the virtual terminals without logging out the process. The /CONTINUE qualifier allows the image PAYROLL.EXE to continue to execute until the process

needs terminal input or attempts to write to the terminal. At that point, the process waits until the physical terminal is reconnected to the virtual terminal. However, you can use the physical terminal to log in again and perform other work.

DISMOUNT

Closes a mounted disk or tape volume for further processing and deletes the logical name associated with the device.

Requires the GRPNAM (group logical name) and SYSNAM (system logical name) privileges to dismount group and system volumes.

Format

DISMOUNT device-name[:]

Parameter

device-name[:]
Name of the device containing the volume—either a logical name or a physical name. If a physical name is specified, the controller defaults to A and the unit defaults to 0.

If the volume currently mounted on the device is a member of a disk or tape volume set, all volumes in the set are dismounted, unless the /UNIT qualifier is specified.

Description

The DISMOUNT command (which invokes the $DISMOU system service) checks for conditions that prevent a Files–11 volume from dismounting. The conditions fall into the following categories:

- Installed swap and page files
- Installed images
- Devices spooled to the volume
- Open user files (any files not falling into one of the first three categories)

If the DISMOUNT command does not find any of these conditions, it performs the following operations:

- Removes the volume from the user's list of mounted volumes, deletes the logical name (if any) associated with the volume, and decrements the mount count.
- If the mount count equals zero after being decremented, the DISMOUNT command marks the volume for dismounting.

  As soon as the volume is idle, that is, after the DISMOUNT command has determined that no user has any open files on the volume, the DISMOUNT command marks a Files–11 volume for dismounting, and dismounts the volume soon.
- If the mount count does not equal zero after being decremented, the DISMOUNT command does not mark the volume for dismount (because the volume must have been mounted shared). In this case, the total effect for the issuing process is that the process is denied access to the volume and the logical name is deleted.

- After a volume is dismounted, nonpaged pool is returned to the system. Paged pool is also returned if the volume was mounted using the /GROUP or /SYSTEM qualifiers.

If the DISMOUNT command does find open files or any other condition that prevents the volume from dismounting, it does *not* mark the volume for dismounting. Instead, the DISMOUNT command displays a message indicating that the volume cannot be dismounted, followed by messages indicating the conditions that exist and the number of instances of each condition.

The /OVERRIDE=CHECKS qualifier allows a volume to be marked for dismounting despite open files or other conditions. For example, marking a volume for dismounting prevents any new files from being opened. Also, when a volume is marked for dismounting, file-system caches are flushed. This activity is especially important when the system is shutting down and the file-system caches must be written to the disk.

If a volume is part of a Files–11 volume set and the /UNIT qualifier is not specified, the entire volume set will be dismounted.

If the volume was mounted with the /SHARE qualifier, it is not actually dismounted until all users who mounted it dismount it or log out. However, the DISMOUNT command deletes the logical name associated with the device.

If the device was allocated with an ALLOCATE command, it remains allocated after the volume is dismounted with the DISMOUNT command. If the device was implicitly allocated by the MOUNT command, the DISMOUNT command deallocates it.

If the volume was mounted with the /GROUP or the /SYSTEM qualifier, it is dismounted even if other users are currently accessing it. The GRPNAM and SYSNAM user privileges are required to dismount group and system volumes, respectively.

Qualifiers

/ABORT

Requires volume ownership or the user privilege VOLPRO (volume protection) to use this qualifier with a volume that was mounted with neither the /GROUP nor the /SYSTEM qualifier. Additionally requires the user privilege SHARE if the volume is mounted privately by a process other than the process issuing the DISMOUNT command.

Specifies that the volume is to be dismounted, regardless of who mounted it. The primary purpose of the /ABORT qualifier is to terminate mount verification. The DISMOUNT/ABORT command also cancels any outstanding I/O requests. If the volume was mounted with the /SHARE qualifier, the /ABORT qualifier causes the volume to be dismounted for all of the users who mounted it.

/CLUSTER

Dismounts a volume throughout a dual-architecture VMScluster system. If you specify DISMOUNT/CLUSTER, the DISMOUNT command checks for open files or other conditions that will prevent a Files–11 volume on the local node from dismounting. If the DISMOUNT command does not find any open files or other conditions, it checks for conditions on all other nodes in the VMScluster. If the DISMOUNT command finds one of the conditions on any node, it displays an

error message identifying the device and the nodes on which the error occurred, followed by an error message indicating open files or other conditions on the volume.

After the DISMOUNT command successfully dismounts the volume on the local node, it dismounts the volume on every other node in the existing VMScluster environment. If the system is not a member of a cluster, the /CLUSTER qualifier has no effect.

/OVERRIDE=CHECKS
Marks a Files–11 volume for dismounting even if files are open on the volume. If you specify DISMOUNT/OVERRIDE=CHECKS, the DISMOUNT command displays messages indicating any open files or other conditions that prevent dismounting, immediately followed by a message indicating that the volume has been marked for dismounting.

A substantial amount of time can pass between the time you enter the DISMOUNT/OVERRIDE=CHECKS command and the completion of the dismount operation. Always wait for the dismount to complete before you remove the volume. (To verify that the dismount has completed, enter the SHOW DEVICES command.) Note that the final phase of volume dismounting occurs in the file system, and all open files on the volume must be closed before the actual dismount can be done. Note also that the file system cannot dismount a volume while any known file lists associated with it contain entries.

/UNIT
Dismounts only one volume of a volume set on the specified device. By default, all volumes in a set are dismounted.

Note

Avoid dismounting the root volume of a volume set, because it contains the master file directory (MFD). It may be impossible to access files on a volume set if the MFD is not accessible.

/UNLOAD
/NOUNLOAD
Determines whether the device on which the volume is mounted is physically unloaded. If you specify the DISMOUNT command without the /UNLOAD or the /NOUNLOAD qualifier, the qualifier that you specified with the MOUNT command (either /UNLOAD or /NOUNLOAD) determines whether the volume is unloaded physically.

Examples

1.
```
$ MOUNT MTA0: PAYVOL TAPE
    .
    .
    .
$ DISMOUNT TAPE
```

The MOUNT command in this example mounts the tape whose volume identification is PAYVOL on the device MTA0: and assigns the logical name TAPE to the device. By default, the volume is not shareable. The

DISMOUNT command releases access to the volume, deallocates the device, and deletes the logical name TAPE.

2.
```
$ MOUNT/SHARE DBA3:  DOC_FILES
    .
    .
    .
$ DISMOUNT DBA3:
```

The MOUNT command in this example mounts the volume labeled DOC_ FILES on the device DBA3. Other users can enter MOUNT commands to access the device. The DISMOUNT command shown in this example deaccesses the device for the process issuing the command. If other users still have access to the volume, the volume remains mounted for their process or processes.

3.
```
$ DISMOUNT/NOUNLOAD  DMA2:
```

The DISMOUNT command in this example dismounts the volume; the /NOUNLOAD qualifier requests that the volume remain in a ready state.

4.
```
$ MOUNT/BIND=PAYROLL  DMA1:,DMA2:  PAYROLL01,PAYROLL02
    .
    .
    .
$ DISMOUNT/UNIT  DMA2:
```

The MOUNT command in this example mounts PAYROLL, a two-volume set. The DISMOUNT command dismounts only PAYROLL02, leaving PAYROLL01 accessible. Note that because the master file directory (MFD) for the volume set is on the root volume, you should not dismount the root volume (in this case, PAYROLL01) of the volume set.

5.
```
$ DISMOUNT $10$DJA100
%DISM-W-CANNOTDMT, $10$DJA100: cannot be dismounted
%DISM-W-INSWPGFIL, 4 swap or page files installed on volume
%DISM-W-SPOOLEDEV, 3 devices spooled to volume
%DISM-W-INSTIMAGE, 7 images installed on volume
%DISM-W-USERFILES, 6 user files open on volume
```

The DISMOUNT command in this example displays the open files and other conditions that prevent device 10DJA100 from dismounting.

6.
```
$ DISMOUNT/CLUSTER $10$DJA100
%DISM-W-RMTDMTFAIL, $10$DJA100: failed to dismount on node SALT
%DISM-W-FILESOPEN, volume has files open on remote node
%DISM-W-RMTDMTFAIL, $10$DJA100: failed to dismount on node PEPPER
%DISM-W-FILESOPEN, volume has files open on remote node
%DISM-W-CANNOTDMT, $10$DJA100: cannot be dismounted
```

The DISMOUNT command in this example displays messages identifying device 10DJA100 and nodes SALT and PEPPER on which errors occurred followed by messages indicating open files on the volume.

DUMP

Displays the contents of a file, a directory, a disk volume, a magnetic tape volume, or a CD–ROM volume in decimal, hexadecimal, octal format, ASCII, or formatted data structures.

Format

DUMP filespec [,...]

Parameter

filespec [,...]

Specifies the file or name of the device being dumped.

If the specified device is not a disk, a tape, or a network device, or if the device is mounted with the /FOREIGN qualifier, the file specification must contain only the device name.

If the specified device is a network device, a disk device, or a tape device that is mounted without the /FOREIGN qualifier, the file specification can contain the asterisk (*) and the percent sign (%) wildcard characters.

Files–11 C/D format standards have been implemented on mounted and foreign mounted volumes.

Description

By default, the DUMP command formats the output both in ASCII characters and in hexadecimal longwords. You can specify another format for the dump by using a radix qualifier (/OCTAL, /DECIMAL, or /HEXADECIMAL) or a length qualifier (/BYTE, /WORD, or /LONGWORD).

Dumping Files

If the input medium is a network device, a disk device, or a tape device that is mounted without the /FOREIGN qualifier, the DUMP command operates on files. You can dump files by either records or blocks. The asterisk (*) and the percent sign (%) wildcard character specifications can be used to select a group of files for processing.

Dumping Volumes

If the input medium is not a disk or a tape device, or if it is mounted with the /FOREIGN qualifier, the DUMP command operates on the input device as a non-file-structured (NFS) medium. Disk devices are dumped by 512-byte logical blocks. Other devices are dumped by physical blocks. No repositioning of the input medium occurs; therefore, consecutive blocks on a tape can be dumped by a single DUMP command.

If you have LOG_IO (logical I/O) privilege, you can dump random blocks on a Files–11 volume. For example, by using the /BLOCKS qualifier, you could dump block 100 on the system disk.

Reading Dumps

The ASCII representation is read left to right. The hexadecimal, decimal, and octal representations are read right to left.

Specifying Numeric Qualifier Values

The numeric values for the /BLOCKS, /RECORDS, and /NUMBER qualifiers can be specified either as decimal numbers or with a leading %X, %O, or %D to signify hexadecimal, octal, or decimal numbers respectively. For example, the following are all valid ways to specify decimal value 24:

```
24
%X18
%O30
%D24
```

Qualifiers

/ALLOCATED

Includes in the dump all blocks allocated to the file. (By default, the dump does not include blocks following the end-of-file [EOF].)

You can specify the /ALLOCATED qualifier if the input is a disk that is mounted without the /FOREIGN qualifier. The /ALLOCATED and /RECORDS qualifiers are mutually exclusive.

/BLOCKS[=(option[,...])]

Dumps the specified blocks one block at a time, which is the default method for all devices except network devices.

Block numbers are specified as integers relative to the beginning of the file. Typically, blocks are numbered beginning with 1. If a disk device is mounted using the /FOREIGN qualifier, blocks are numbered beginning with zero. Select a range of blocks to be dumped by specifying one of the following options:

START:*n*	Specifies the number of the first block to be dumped; the default is the first block.
END:*n*	Specifies the number of the last block to be dumped; the default is the last block or the end-of-file (EOF) block, depending on whether you have specified the /ALLOCATED qualifier.
COUNT:*n*	Specifies the number of blocks to be dumped. The COUNT option provides an alternative to the END option; you cannot specify both.

If you specify only one option, you can omit the parentheses.

The /BLOCKS and /RECORDS qualifiers are mutually exclusive.

Use the /BLOCKS qualifier to dump random blocks from Files–11 volumes. This procedure requires LOG-IO (logical I/O) privilege.

/BYTE

Formats the dump in bytes. The /BYTE, /LONGWORD, and /WORD qualifiers are mutually exclusive. The default format is composed of longwords.

/DECIMAL

Dumps the file in decimal radix. The /DECIMAL, /HEXADECIMAL (default), and /OCTAL qualifiers are mutually exclusive.

/DESCRIPTOR[=(option[,...])]

Dumps the specified ISO 9660 volume descriptors in a formatted manner. If /NOFORMATTED is specified, block mode format is used.

The descriptor options that you can specify are as follows:

BOOT:*n*	Searches for the *n*th occurrence of a Boot Record.
PVD:*n*	Searches for the *n*th occurrence of a Primary Volume Descriptor.
SVD:*n*	Searches for the *n*th occurrence of a Supplementary Volume Descriptor.
VPD:*n*	Searches for the *n*th occurrence of a Volume Partition Descriptor.
VDST:*n*	Searches for the *n*th occurrence of a Volume Descriptor Set Terminator.

If you specify only one option, you can omit the parentheses.

ISO 9660 descriptors are specified by their ordinal position from the start of the volume, defaulting to 1 if they are not specified. The ISO 9660 volume is sequentially searched from the beginning of the volume descriptor set sequence to the end to find the specified descriptor and output it in a formatted manner.

/DIRECTORY
Dumps data blocks of the specified file as formatted on-disk structures of either Files–11 On-Disk Structure Level 1 or On-Disk Structure Level 2 directory records, ISO 9660, or High Sierra directory records.

/EXACT
Use with the /PAGE=SAVE and /SEARCH qualifiers to specify a search string that must match the search string exactly and must be enclosed with quotation marks (‘‘ ’’).

If you specify the /EXACT qualifier without the /SEARCH qualifier, exact search mode is enabled when you set the search string with the Find (E1) key.

/FILE_HEADER
Dumps each data block that is a valid Files–11 header in Files–11 header format rather than in the selected radix and length formats.

/FORMATTED (default)
/NOFORMATTED
Dumps the file header in Files–11 format; the /NOFORMATTED qualifier dumps the file header in octal format. This qualifier is useful only when the /HEADER qualifier is specified.

/HEADER
Dumps the file header and access control list (ACL). To dump only the file header, and not the file contents, also specify /BLOCK=(COUNT:0). The /HEADER qualifier is invalid for devices mounted using the /FOREIGN qualifier.

Use the /FORMATTED qualifier to control the format of the display.

You can use the /FILE_HEADER qualifier with the /HEADER qualifier to have Files–11 file headers printed in an interpreted representation.

By default, the file header is not displayed.

/HEXADECIMAL (default)
Dumps the file in hexadecimal radix. The /DECIMAL, /HEXADECIMAL (default), and /OCTAL qualifiers are mutually exclusive.

/HIGHLIGHT[=keyword]
/NOHIGHLIGHT (default)
Use with the /PAGE=SAVE and /SEARCH qualifiers to specify the type of highlighting you want when a search string is found. When a string is found, the entire line is highlighted. You can use the following keywords: BOLD, BLINK, REVERSE, and UNDERLINE. BOLD is the default highlighting.

/IDENTIFIER=file-id
Dumps the file selected by the file identification (FID) number from the specified volume. For further information, see the /FILE_ID qualifier from the DCL command, DIRECTORY.

/LONGWORD (default)
Formats the dump in longwords. The /BYTE, /LONGWORD, and /WORD qualifiers are mutually exclusive.

/MEDIA_FORMAT=keyword
Specifies the format in which a data structure is to be dumped. If you specify this qualifier, you must use one of the following keywords:

CDROM	Specifies ISO 9660 media format. This format is the default if you do not specify the /MEDIA_FORMAT qualifier.
CDROM_HS	Specifies High Sierra media format.

/NUMBER[=n]
Specifies how byte offsets are assigned to the lines of output. If you specify the /NUMBER qualifier, the byte offsets increase continuously through the dump, beginning with *n*; if you omit the /NUMBER qualifier, the first byte offset is zero. By default, the byte offset is reset to zero at the beginning of each block or record.

/OCTAL
Dumps the file in octal radix. The /DECIMAL, /HEXADECIMAL (default), and /OCTAL qualifiers are mutually exclusive.

/OUTPUT[=filespec]
Specifies the output file for the dump. If you do not specify a file specification, the default is the file name of the file being dumped and the file type .DMP. If the /OUTPUT qualifier is not specified, the dump goes to SYS$OUTPUT. The /OUTPUT and /PRINTER qualifiers are mutually exclusive.

/PAGE[=keyword]
/NOPAGE (default)
Controls the display of dump information on the screen.

You can use the following keywords with the /PAGE qualifier:

CLEAR_SCREEN	Clears the screen before each page is displayed.
SCROLL	Displays information one line at a time.
SAVE[=*n*]	Enables screen navigation of information, where *n* is the number of pages to store.

The /PAGE=SAVE qualifier allows you to navigate through screens of information. The /PAGE=SAVE qualifier stores up to 5 screens of up to 255 columns of information. When you use the /PAGE=SAVE qualifier, you can use the following keys to navigate through the information:

Key Sequence	Description
Up arrow (↑), Ctrl/B	Scroll up one line.
Down arrow (↓)	Scroll down one line.
Left arrow (←)	Scroll left one column.
Right arrow (→)	Scroll right one column.
Find (E1)	Specify a string to find when the information is displayed.
Insert Here (E2)	Scroll right one half screen.
Remove (E3)	Scroll left one half screen.
Select (E4)	Toggle 80/132 column mode.
Prev Screen (E5)	Get the previous page of information.
Next Screen (E6), Return, Enter, Space	Get the next page of information.
F10, Ctrl/Z	Exit. (Some utilities define these differently.)
Help (F15)	Display utility help text.
Do (F16)	Toggle the display to oldest/newest page.
Ctrl/W	Refresh the display.

The /PAGE qualifier is not compatible with the /OUTPUT qualifier.

/PATH_TABLE

Dumps data blocks in ISO 9660 Path Table format.

/PRINTER

Queues the dump to SYS$PRINT in a file named with the file name of the file being dumped and the file type .DMP. If the /PRINTER qualifier is not specified, the dump goes to SYS$OUTPUT. The asterisk (*) and the percent sign (%) wildcard characters are not allowed. The /OUTPUT and /PRINTER qualifiers are mutually exclusive.

/RECORDS[=(option[,...])]

Dumps the file a record at a time rather than a block at a time. (By default, input is dumped one block at a time for all devices except network devices.)

Records are numbered beginning with 1.

Select a range of records to be dumped by specifying one of the following options:

START:*n*	Specifies the number of the first record to be dumped; the default is the first record.
END:*n*	Specifies the number of the last record to be dumped; the default is the last record of the file.
COUNT:*n*	Specifies the number of records to be dumped. The COUNT option provides an alternative to the END option; you cannot specify both.

If you specify only one option, you can omit the parentheses.

If you specify the /RECORDS qualifier, you cannot specify the /ALLOCATED or the /BLOCKS qualifier.

/SEARCH="string"
Use with the /PAGE=SAVE qualifier to specify a string that you want to find in the information being displayed. Quotation marks are required for the /SEARCH qualifier, if you include spaces in the text string.

You can also dynamically change the search string by pressing the Find key (E1) while the information is being displayed. Quotation marks are not required for a dynamic search.

/VALIDATE_HEADER
Verifies /DIRECTORY records for Files–11.

/WIDTH=n
Formats the dump output into 80 or 132 columns by specifying *n* as either 80 or 132.

/WORD
Formats the dump in words. The /BYTE, /LONGWORD, and /WORD qualifiers are mutually exclusive.

/WRAP
/NOWRAP (default)
Use with the /PAGE=SAVE qualifier to limit the number of columns to the width of the screen and to wrap lines that extend beyond the width of the screen to the next line.

The /NOWRAP qualifier extends lines beyond the width of the screen and can be seen when you use the scrolling (left and right) features provided by the /PAGE=SAVE qualifier.

Examples

1.
```
$ DUMP TEST.DAT
Dump of file DISK0:[NORMAN]TEST.DAT;1 on 14-DEC-1994 15:43:26.08
File ID (3134,818,2)   End of file block 1 / Allocated 3
Virtual block number 1 (00000001), 512 (0200) bytes
 706D6173 20612073 69207369 68540033 3.This is a samp 000000
 73752065 62206F74 20656C69 6620656C le file to be us 000010
 61786520 504D5544 2061206E 69206465 ed in a DUMP exa 000020
 00000000 00000000 0000002E 656C706D mple............ 000030
 00000000 00000000 00000000 00000000 ................ 000040
 00000000 00000000 00000000 00000000 ................ 000050
 00000000 00000000 00000000 00000000 ................ 000060
          .
          .
          .
 00000000 00000000 00000000 00000000 ................ 0001E0
 00000000 00000000 00000000 00000000 ................ 0001F0
```

The DUMP command displays the contents of TEST.DAT both in hexadecimal longword format and in ASCII beginning with the first block in the file.

2.
```
$ DUMP TEST.DAT/OCTAL/BYTE
Dump of file DISK0:[NORMAN]TEST.DAT;1 on 14-DEC-1994 15:45:33.58
File ID (74931,2,1)   End of file block 1 / Allocated 3
Virtual block number 1 (00000001), 512 (0200) bytes
 151 040 163 151 150 124 000 063 3.This i 000000
 160 155 141 163 040 141 040 163 s a samp 000010
 040 145 154 151 146 040 145 154 le file  000020
 163 165 040 145 142 040 157 164 to be us 000030
 040 141 040 156 151 040 144 145 ed in a  000040
 141 170 145 040 120 115 125 104 DUMP exa 000050
 377 377 000 056 145 154 160 155 mple.... 000060
 000 000 000 000 000 000 000 000 ........ 000070
 000 000 000 000 000 000 000 000 ........ 000100
 000 000 000 000 000 000 000 000 ........ 000110
          .
          .
          .
 000 000 000 000 000 000 000 000 ........ 000760
 000 000 000 000 000 000 000 000 ........ 000770
```

The DUMP command displays the image of the file TEST.DAT, formatted both in octal bytes and in ASCII characters beginning with the first block.

3.
```
$ DUMP NODE3::DISK2:[STATISTICS]RUN1.DAT
```

This command line dumps the file RUN1.DAT that is located at remote node NODE3. The default DUMP format will be used.

4.
```
$ DUMP/HEADER/BLOCK=COUNT=0 SYS$SYSTEM:DATASHARE.EXE

Dump of file SYS$SYSTEM:DATASHARE.EXE on  12-NOV-1994 16:06:46.75
File ID (16706,59,0)   End of file block 410 / Allocated 411

                            File Header

Header area
    Identification area offset:           40
    Map area offset:                      100
    Access control area offset:           255
    Reserved area offset:                 255
    Extension segment number:             0
    Structure level and version:          2, 1
    File identification:                  (16706,59,0)
    Extension file identification:        (0,0,0)
    VAX RMS attributes
        Record type:                      Fixed
        File organization:                Sequential
        Record attributes:                <none specified>
        Record size:                      512
        Highest block:                    411
        End of file block:                410
        End of file byte:                 414
        Bucket size:                      0
        Fixed control area size:          0
        Maximum record size:              512
        Default extension size:           0
        Global buffer count:              0
        Directory version limit:          0
    File characteristics:                 Contiguous best try
    Map area words in use:                3
    Access mode:                          0
    File owner UIC:                       [1,4]
    File protection:                      S:RWED, O:RWED, G:RE, W:
    Back link file identification:        (7149,80,0)
    Journal control flags:                <none specified>
    Active recovery units:                None
```

```
    Highest block written:               411

Identification area
    File name:                           DATASHARE.EXE
    Revision number:                     1
    Creation date:                       12-AUG-1994 14:06:49.84
    Revision date:                       12-AUG-1994 14:06:53.20
    Expiration date:                     <none specified>
    Backup date:                         <none specified>

Map area
    Retrieval pointers
        Count:        411        LBN:    1297155

Checksum:                                30710
```

In this example, the DUMP command dumps the file header of the specified file. Since this file is recorded on Files–11 ODS-2 9660 media, the file header is displayed in a Files–11 File Header format. Imbedded on the Files–11 Header is a VAX RMS attributes block.

5.
```
$ DUMP/HEADER/BLOCK=COUNT=0 DISK$GRIPS_2:[000000]AAREADME.TXT;
Dump of file DISK$GRIPS_2:[000000]AAREADME.TXT;1 on 15-DEC-1994
10:07:29.70

   File ID (4,6,0)   End of file block 29 / Allocated 29

                           ISO 9660 File Header

   Length of Directory Record:              48
   Extended Attribute Length:               1
   Location of Extent (LSB/MSB):            312/312
   Data Length of File Section (LSB/MSB):   14640/14640
   Recording Date and Time                  10-DEC-1994 16:22:30 GMT(0)
   File Flags                               RECORD, PROTECTION
   Interleave File Unit size:               0
   Interleave Gap size:                     0
   Volume Sequence # of extent (LSB/MSB):   1/1
   File Identifier Field Length:            14
   File Identifier:                         AAREADME.TXT;1
   System Use
5458542E 454D4441 45524141 0E010000 01000018 001E1610 100B5930 39000000
...90Y.............AAREADME.TXT 000000
                                                                    00313B
;1............................. 000020

Extended Attribute record
   Owner Identification (LSB/MSB):          7/7
   Group Identification (LSB/MSB):          246/246
   Access permission for classes of users   S:R, O:R, G:RE, W:RE
   File Creation Date/Time:                 5-OCT-1994 14:17:49.29 GMT(0)
   File Modification Date/Time:             6-NOV-1994 16:22:30.96 GMT(0)
   File Expiration Date/Time:               00-00-0000 00:00:00.00 GMT(0)
   File Effective Date/Time:                00-00-0000 00:00:00.00 GMT(0)
   Record Format                            Fixed
   Record Attributes                        CRLF
   Record Length (LSB/MSB):                 80/80
   System Identifier:
   System Use
   Extended Attribute Version:              1
   Escape Sequence record length:           0
   Application Use Length (LSB/MSB):        0/0
   Application Use
```

```
VAX RMS attributes
    Record type:                    Fixed
    File organization:              Sequential
    Record attributes:              Implied carriage control
    Record size:                    80
    Highest block:                  29
    End of file block:              29
    End of file byte:               304
    Bucket size:                    0
    Fixed control area size:        0
    Maximum record size:            80
    Default extension size:         0
    Global buffer count:            0
    Directory version limit:        0
```

The DUMP/HEADER command dumps the file header of the specified file. Since this file is recorded on ISO 9660 media, the file header is displayed in the format of an ISO 9660 File Header, and, since this file contains an optional ISO 9660 Extended Attribute Record (XAR), it is also displayed. Finally, as with all DUMP/HEADER requests, VAX RMS attributes are displayed.

EDIT/ACL

Invokes the access control list (ACL) editor, which creates or modifies an access control list for a specified object. The /ACL qualifier is required. For a complete description of the ACL Editor, see the *OpenVMS System Management Utilities Reference Manual* or the *OpenVMS Guide to System Security*.

Format

EDIT/ACL object-spec

EDIT/EDT

Invokes EDT, an interactive text editor. The /EDT qualifier is required.

Format

EDIT/EDT filespec

Parameter

filespec
Specifies the file to be created or edited using EDT. If the file does not exist, it is created by EDT.

EDT does not provide a default file type when creating files; if you do not include a file type, it is null. The file must be a disk file on a Files–11 formatted volume.

The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

Description

EDT creates or edits text files. You can use EDT to enter or edit text in three modes: keypad, line, or nokeypad. Keypad editing, which is screen-oriented, is available on VT300-series, VT200-series, VT100, and VT52 terminals. A screen-oriented editor allows you to see several lines of text at once and move the cursor throughout the text in any direction. Line editing operates on all terminals. In fact, if you have a terminal other than a VT300-series, VT200-series, VT100, or VT52, line editing is the only way you can use EDT. You might prefer line editing if you are accustomed to editing by numbered lines. Nokeypad mode is a command-oriented screen editor available on VT300-series, VT200-series, VT100, and VT52 terminals. You can use line mode and nokeypad mode to redefine keys for use in keypad mode.

When you invoke EDT, you are in line mode by default. If you are editing an existing file, EDT displays the line number and text for the first line of the file. If you are creating a new file, EDT displays the following message:

```
Input file does not exist
[EOB]
```

In either case, EDT then displays the line mode prompt, which is the asterisk (*).

For complete details on the EDT editor, see the *OpenVMS EDT Reference Manual*.

Qualifiers

/COMMAND[=filespec]
/NOCOMMAND
Determines whether or not EDT uses a startup command file. The /COMMAND file qualifier should be followed by an equal sign (=) and the specification of the command file. The default file type for command files is .EDT. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

The following command line invokes EDT to edit a file named MEMO.DAT and specifies that EDT use a startup command file named XEDTINI.EDT:

```
$ EDIT/COMMAND=XEDTINI.EDT MEMO.DAT
```

If you do not include the /COMMAND=command file qualifier, EDT looks for the EDTSYS logical name assignment. If EDTSYS is not defined, EDT processes the systemwide startup command file SYS$LIBRARY:EDTSYS.EDT. If this file does not exist, EDT looks for the EDTINI logical name assignment. If EDTINI is not defined, EDT looks for the file named EDTINI.EDT in your default directory. If none of these files exists, EDT begins your editing session in the default state.

To prevent EDT from processing either the systemwide startup command file or the EDTINI.EDT file in your default directory, use the /NOCOMMAND qualifier as follows:

```
$ EDIT/EDT/NOCOMMAND MEMO.DAT
```

/CREATE (default)
/NOCREATE

Controls whether EDT creates a new file when the specified input file is not found.

Normally, EDT creates a new file to match the input file specification if it cannot find the requested file name in the specified directory. When you use the /NOCREATE qualifier in the EDT command line and type a specification for a file that does not exist, EDT displays an error message and returns to the DCL command level as follows:

```
$ EDIT/EDT/NOCREATE NEWFILE.DAT
Input file does not exist
$
```

/JOURNAL[=journal-file]
/NOJOURNAL

Determines whether EDT keeps a journal during your editing session. A journal contains a record of the keystrokes you enter during an editing session. The default file name for the journal is the same as the input file name. The default file type is .JOU. The /JOURNAL qualifier enables you to use a different file specification for the journal.

The following command line invokes EDT to edit a file named MEMO.DAT and specifies the name SAVE.JOU for the journal:

```
$ EDIT/EDT/JOURNAL=SAVE MEMO.DAT
```

If you are editing a file from another directory and want the journal to be located in that directory, you must use the /JOURNAL qualifier with a file specification that includes the directory name. Otherwise, EDT creates the journal in the default directory.

The directory that is to contain the journal should not be write-protected.

To prevent EDT from keeping a record of your editing session, use the /NOJOURNAL qualifier in the EDT command line as follows:

```
$ EDIT/EDT/NOJOURNAL MEMO.DAT
```

Once you have created a journal, enter the EDT/RECOVER command to execute the commands in the journal. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

/OUTPUT=output-file
/NOOUTPUT

Determines whether EDT creates an output file at the end of your editing session. The default file specification for both the input file and the output file is the same. Use the /OUTPUT qualifier to give the output file a different file specification from the input file.

The following command line invokes EDT to edit a file named MEMO.DAT and gives the resulting output file the name OUTMEM.DAT:

```
$ EDIT/EDT/OUTPUT=OUTMEM.DAT MEMO.DAT
```

You can include directory information as part of your output file specification to send output to another directory as follows:

```
$ EDIT/EDT/OUTPUT=[BARRETT.MAIL]MEMO.DAT MEMO.DAT
```

The /NOOUTPUT qualifier suppresses the creation of an output file, but not the creation of a journal. If you decide that you do not want an output file, you can use the /NOOUTPUT qualifier as follows:

```
$ EDIT/EDT/NOOUTPUT MEMO.DAT
```

A system interruption does not prevent you from re-creating your editing session because a journal is still being maintained. To save your editing session, even when you specify /NOOUTPUT, use the line mode command WRITE to put the text in an external file before you end the session.

The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

/READ_ONLY
/NOREAD_ONLY (default)

Determines whether EDT keeps a journal and creates an output file. With the /NOREAD_ONLY qualifier, EDT maintains the journal and creates an output file when it processes the line mode command EXIT. Using the /READ_ ONLY qualifier has the same effect as specifying both the /NOJOURNAL and /NOOUTPUT qualifiers.

The following command line invokes EDT to edit a file named CALENDAR.DAT, but does not create a journal or an output file:

```
$ EDIT/EDT/READ_ONLY CALENDAR.DAT
```

Use the /READ_ONLY qualifier when you are searching a file and do not intend to make any changes to it. To modify the file, use the line mode command WRITE to save your changes. Remember, however, that you have no journal.

/RECOVER
/NORECOVER (default)

Determines whether EDT reads a journal at the start of the editing session.

When you use the /RECOVER qualifier, EDT reads the appropriate journal and processes whatever commands it contains. The appropriate syntax is as follows:

```
$ EDIT/EDT/RECOVER MEMO.DAT
```

If the journal file type is not .JOU or the file name is not the same as the input file name, you must include both the /JOURNAL qualifier and the /RECOVER qualifier as follows:

```
$ EDIT/EDT/RECOVER/JOURNAL=SAVE.XXX MEMO.DAT
```

Because the /NORECOVER qualifier is the default for EDT, you do not need to specify it in a command line.

Examples

1.
```
$ EDIT/EDT/OUTPUT=NEWFILE.TXT OLDFILE.TXT
    1       This is the first line of the file OLDFILE.TXT.
*
```

   This command invokes EDT to edit the file OLDFILE.TXT. EDT looks for the EDTSYS logical name assignment. If EDTSYS is not defined, EDT processes the systemwide startup command file SYS$LIBRARY:EDTSYS.EDT. If this file does not exist, EDT looks for the EDTINI logical name assignment. If EDTINI is not defined, EDT looks for the file named EDTINI.EDT in your default directory. If none of these files exists, EDT begins your editing session in the default state. When the session ends, the edited file has the name NEWFILE.TXT.

2.
```
$ EDIT/EDT/RECOVER OLDFILE.TXT
```

   This command invokes EDT to recover from an abnormal exit during a previous editing session. EDT opens the file OLDFILE.TXT, and then processes the journal OLDFILE.JOU. Once the journal has been processed, the user can resume interactive editing.

EDIT/FDL

Invokes the Edit/FDL (File Definition Language) utility, which creates and modifies FDL files. The /FDL qualifier is required. For a complete description of the File Definition Language utility, see the *OpenVMS Record Management Utilities Reference Manual*.

Format

EDIT/FDL filespec

EDIT/SUM

Invokes the SUMSLP utility, a batch-oriented editor, to update a single input file with multiple files of edit commands.

For a complete description of the SUMSLP utility, see the *OpenVMS SUMSLP Utility Manual*.

Format

EDIT/SUM input-file

EDIT/TECO

Invokes the TECO interactive text editor.

Format

EDIT/TECO [filespec]

EDIT/TECO/EXECUTE=command-file [argument]

Parameter

filespec
Specifies the file to be created or edited using the TECO editor. If the file does not exist, it is created by TECO, unless you specify the /NOCREATE qualifier. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

If you specify the /MEMORY qualifier (default) without a file specification, TECO edits the file identified by the logical name TEC$MEMORY. If TEC$MEMORY has no equivalence string, or if the /NOMEMORY qualifier is specified, TECO starts in command mode and does not edit an existing file.

If you specify the /MEMORY qualifier and a file specification, the file specification is equated to the logical name TEC$MEMORY.

argument
See the /EXECUTE qualifier.

Description

The TECO editor creates or edits text files. For detailed information on the use of TECO, see the *PDP–11 TECO Editor Reference Manual*.

Qualifiers

/COMMAND[=filespec]
/NOCOMMAND
Controls whether a startup command file is used. The /COMMAND file qualifier may be followed by an equal sign (=) and the specification of the command file. The default file type for command files is .TEC.

The following command line invokes TECO to edit a file named MEMO.DAT and specifies that TECO use a startup command file named XTECOINI.TEC:

```
$ EDIT/TECO/COMMAND=XTECOINI.TEC MEMO.DAT
```

If you do not include the /COMMAND qualifier, or if you enter /COMMAND without specifying a command file, TECO looks for the TEC$INIT logical name assignment. If TEC$INIT is not defined, no startup commands are executed.

The logical name TEC$INIT can equate either to a string of TECO commands or to a dollar sign ($) followed by a file specification. If TEC$INIT translates to a string of TECO commands, the string is executed; if it translates to a dollar sign followed by a file specification, the contents of the file are executed as a TECO command string. For further information, see the *PDP–11 TECO Editor Reference Manual*.

To prevent TECO from using any startup command file, use the /NOCOMMAND qualifier as follows:

```
$ EDIT/TECO/NOCOMMAND MEMO.DAT
```

The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

/CREATE (default)
/NOCREATE
Creates a new file when the specified input file cannot be found. If the /MEMORY qualifier is specified and no input file is specified, the file created is the one specified by the logical name TEC$MEMORY. Normally, TECO creates a new file to match the input file specification if it cannot find the requested file name in the specified directory. When you use the /NOCREATE qualifier in the TECO command line and type a specification for a file that does not exist, TECO displays an error message and returns you to the DCL command level. The /CREATE and /NOCREATE qualifiers are incompatible with the /EXECUTE qualifier.

/EXECUTE=command-file [argument]
Invokes TECO and executes the TECO macro found in the command file. The argument, if specified, appears in the text buffer when macro execution starts. Blanks or special characters must be enclosed in quotation marks (‘‘ ’’). For detailed information on the use of TECO macros, see the *PDP–11 TECO Editor Reference Manual*.

The /EXECUTE qualifier is incompatible with the /CREATE and /MEMORY qualifiers.

/MEMORY (default)
/NOMEMORY
Specifies that the last file you edited with TECO, identified by the logical name TEC$MEMORY, will be the file edited if you omit the file specification to the EDIT/TECO command.

/OUTPUT=output-file
/NOOUTPUT (default)
Controls how the output file is named at the end of your editing session. By default, the output file has the same name as the input file but is given the next higher available version number. Use the /OUTPUT qualifier to give the output file a file specification different from the input file.

The following command line invokes TECO to edit a file named MEMO.DAT and gives the resulting output file the name OUTMEM.DAT:

```
$ EDIT/TECO/OUTPUT=OUTMEM.DAT MEMO.DAT
```

You can include directory information as part of your output file specification to send output to another directory as follows:

```
$ EDIT/TECO/OUTPUT=[BARRRET.MAIL]MEMO.DAT MEMO.DAT
```

The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

/READ_ONLY
/NOREAD_ONLY (default)
Controls whether an output file is created. By default, an output file is created; the /READ_ONLY qualifier suppresses the creation of the output file.

Examples

1. `$ EDIT/TECO/OUTPUT=NEWFILE.TXT OLDFILE.TXT`

   This EDIT command invokes the TECO editor to edit the file OLDFILE.TXT. TECO looks for the TEC$INIT logical name assignment. If TEC$INIT is not defined, TECO begins the editing session without using a command file. When the session ends, the edited file has the name NEWFILE.TXT.

2. `$ EDIT/TECO/EXECUTE=FIND_DUPS "TEMP, ARGS, BLANK"`

   In this example, the /EXECUTE qualifier causes the TECO macro contained in the file FIND_DUPS.TEC to be executed, with the argument string ‘‘TEMP, ARGS, BLANK’’ located in the text buffer.

EDIT/TPU

Invokes the DEC Text Processing utility (DECTPU). By default, this runs the Extensible Versatile Editor (EVE). DECTPU provides a structured programming language and other components for creating text editors and other applications. EVE is a general-purpose text editor that is the OpenVMS default editor.

For information about editing with EVE, see the *OpenVMS User's Manual* and online help.

Format

EDIT[/TPU] [input-file]

ENABLE AUTOSTART

Enables the autostart feature on a node for all autostart queues managed by the specified queue manager. By default, this command uses the /QUEUES qualifier.

Requires OPER (operator) privileges.

For more information on autostart queues, see the chapter on batch and print queues in the *OpenVMS System Manager's Manual*.

Format

ENABLE AUTOSTART[/QUEUES]

Parameters

None.

Description

Enabling autostart for queues notifies the queue manager to automatically start all of its stopped active autostart queues on a node. It also notifies the queue manager to automatically start any of its autostart queues that fail over to the node. By default, the ENABLE AUTOSTART command affects the node from which it is entered. Specify the /ON_NODE qualifier to enable autostart on a different node.

By default, the command affects autostart queues managed by the default queue manager, SYS$QUEUE_MANAGER. Specify the /NAME_OF_MANAGER qualifier to disable autostart of a different queue manager's autostart queues on the node.

An autostart queue is active if it has been activated by the /START qualifier with the INITIALIZE/QUEUE command or by the START/QUEUE command and has not been stopped by the STOP/QUEUE/NEXT or STOP/QUEUE/RESET command.

When a node boots, autostart is disabled until you enter the ENABLE AUTOSTART command. Typically, you would add this command to your site-specific startup command procedure or your queue startup command procedure to start a node's autostart queues each time the node boots.

Qualifiers

/NAME_OF_MANAGER=name

Specifies the name of the queue manager controlling the autostart queues you want to enable. The qualifier allows the autostart feature to be used differently for different sets of queues.

If the /NAME_OF_MANAGER qualifier is omitted, the default queue manager name SYS$QUEUE_MANAGER is used.

For more information on multiple queue managers, see the chapter on the queue manager in the *OpenVMS System Manager's Manual*.

/ON_NODE=nodename

Specifies a node in a VMScluster system. Use this qualifier to enable autostart on a node other than the one from which you enter the command.

/QUEUES
Specifies that autostart is to be enabled for queues. (This qualifier is used by default.)

Examples

1.
```
$ INITIALIZE/QUEUE/BATCH/START-
_$ /AUTOSTART_ON=SATURN:: BATCH_1
$ ENABLE AUTOSTART/QUEUES
   .
   .
   .
$ DISABLE AUTOSTART/QUEUES
```

In this example, the INITIALIZE/QUEUE command creates an autostart queue BATCH_1, capable of running on node SATURN. The /START qualifier activates the queue for autostart. The ENABLE/AUTOSTART/QUEUES command (executed on node SATURN) enables autostart on the node, causing the queue (and any other active autostart queues on the node) to begin executing jobs.

The DISABLE AUTOSTART command (executed on node SATURN) stops autostart queues on the node and prevents any queues from failing over to the node.

These commands only affect queues managed by the default queue manager SYS$QUEUE_MANAGER because the /NAME_OF_MANAGER qualifier is not specified.

Because BATCH_1 is set up to run only on one node, the queue cannot fail over to another node and therefore is stopped. However, the queue remains active for autostart and will be started when the ENABLE AUTOSTART command is entered for node SATURN. No START/QUEUE command is needed to restart BATCH_1 unless autostart of the queue is deactivated with the STOP/QUEUE/NEXT or STOP/QUEUE/RESET command.

2.
```
$ INITIALIZE/QUEUE/BATCH/START-
_$ /AUTOSTART_ON=(NEPTUN::,SATURN::) BATCH_1
$ ENABLE AUTOSTART/QUEUES/ON_NODE=NEPTUN
$ ENABLE AUTOSTART/QUEUES/ON_NODE=SATURN
   .
   .
   .
$ STOP/QUEUES/ON_NODE=NEPTUN
```

In this example, the INITIALIZE/QUEUE command creates an autostart queue BATCH_1. The /START qualifier activates the queue for autostart.

The first ENABLE AUTOSTART/QUEUES command causes the queue to begin executing on node NEPTUN. The second ENABLE AUTOSTART /QUEUES command enables autostart on node SATURN to start all stopped active autostart queues on that node and to start any autostart queues that might fail over to that node.

Later, suppose node NEPTUN must be removed from the VMScluster system. The STOP/QUEUES/ON_NODE command stops all queues on node NEPTUN, and causes the autostart queue BATCH_1 to fail over to node SATURN. Because the queue is active for autostart, and because autostart has been enabled on node SATURN, the queue is automatically started on that node.

This command only affects queues managed by the default queue manager SYS$QUEUE_MANAGER because the /NAME_OF_MANAGER qualifier is not specified.

ENDSUBROUTINE

Defines the end of a subroutine in a command procedure. For more information about the ENDSUBROUTINE command, refer to the description of the CALL command.

Format

ENDSUBROUTINE

EOD

Signals the end of a data stream when a command or program is reading data from an input device other than an interactive terminal.

Format

$ EOD

Parameters

None.

Description

The EOD (end of deck) command in a command procedure or in a batch job does the following:

- Terminates input data lines that begin with dollar signs ($). The DECK command indicates that the following lines begin with dollar signs and should be interpreted as data, not as commands; the EOD command indicates the end of the data lines.
- Terminates an input file if multiple input files are contained in the command stream without intervening commands. The program or command reading the data receives an end-of-file (EOF) condition when the EOD command is read.

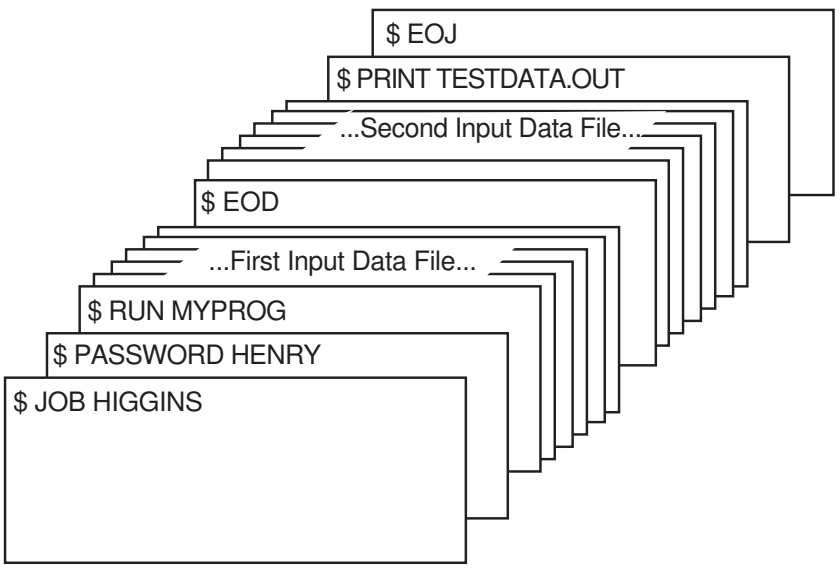
The EOD command must be preceded by a dollar sign; the dollar sign must be in the first character position (column 1) of the input record.

Examples

1.
```
$ CREATE WEATHER.COM
$ DECK
$ FORTRAN WEATHER
$ LINK WEATHER
$ RUN WEATHER
$ EOD
$ @WEATHER
```

In this example, the command procedure creates a command procedure called WEATHER.COM. The lines delimited by the DECK and EOD commands are written to the file WEATHER.COM. Then the command procedure executes WEATHER.COM.

2.



ZK–0785–GE

The program MYPROG requires two input files; these are read from the logical device SYS$INPUT. The EOD command signals the end of the first data file and the beginning of the second. The next line that begins with a dollar sign (a PRINT command in this example) signals the end of the second data file.

EOJ

Marks the end of a batch job submitted through a card reader.

Format

$ EOJ

Parameters

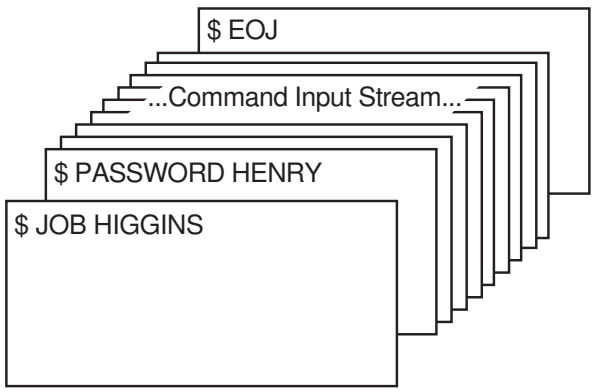
None.

Description

The EOJ (end of job) command marks the end of a batch job submitted through a card reader. An EOJ card is not required; however, if present, the first nonblank character in the command line must be a dollar sign ($). If issued in any other context, the EOJ command logs the process out. The EOJ command cannot be abbreviated.

The EOF card is equivalent to the EOJ card.

Example



ZK–0786–GE

The JOB and PASSWORD commands mark the beginning of a batch job submitted through the card reader; the EOJ command marks the end of the job.

EXAMINE

Displays the contents of virtual memory.

Requires user-mode read (R) access to the virtual memory location whose contents you want to examine.

Format

EXAMINE location[:location]

Parameter

location[:location]

Specifies a virtual address or a range of virtual addresses (where the second address is larger than the first) whose contents you want to examine. If you specify a range of addresses, separate the beginning and ending addresses with a colon (:).

A location can be any valid arithmetic expression containing arithmetic or logical operators or previously assigned symbols. Radix qualifiers determine the radix in which the address is interpreted; hexadecimal is the initial default radix. Symbol names are always interpreted in the radix in which they were defined. The radix operators %X, %D, or %O can precede the location. A hexadecimal value must begin with a number (or be preceded by %X).

The DEPOSIT and EXAMINE commands maintain a pointer to the current memory location. The EXAMINE command sets this pointer to the last location examined when you specify an EXAMINE command. You can refer to this location using the period (.) in a subsequent EXAMINE command or DEPOSIT command.

Description

The EXAMINE command displays the contents of virtual memory. The address is displayed in hexadecimal format and the contents are displayed in the radix requested, as follows:

```
address:  contents
```

If the address specified is not accessible to user mode, four asterisks (*) are displayed in the contents field.

Radix Qualifiers: The radix default for a DEPOSIT command or an EXAMINE command determines how the command interprets numeric literals. The initial default radix is hexadecimal; all numeric literals in the command line are assumed to be hexadecimal values. If a radix qualifier modifies an EXAMINE command, that radix becomes the default for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it. For example:

```
$ EXAMINE/DECIMAL 900
00000384:  0554389621
```

The EXAMINE command interprets the location 900 as a decimal number and displays the contents of that location in decimal. All subsequent DEPOSIT and EXAMINE commands assume that numbers you enter for addresses and data are decimal. Note that the EXAMINE command always displays the address location in hexadecimal format.

Symbol names defined by = (assignment statement) commands are always interpreted in the radix in which they were defined.

Note that hexadecimal values entered as examine locations or as data to be deposited must begin with a numeric character (0 to 9). Otherwise, the command interpreter assumes that you have entered a symbol name, and attempts symbol substitution.

You can use the radix operators %X, %D, or %O to override the current default when you enter the EXAMINE command. For example:

```
$ EXAMINE/DECIMAL %X900
00000900:  321446536
```

This command requests a decimal display of the data in the location specified as hexadecimal 900.

Length Qualifiers: The initial default length unit for the EXAMINE command is a longword. The EXAMINE command displays data, 1 longword at a time, with blanks between longwords. If a length qualifier modifies the command, that length becomes the default length of a memory location for subsequent EXAMINE and DEPOSIT commands, until another qualifier overrides it.

Restriction on Placement of Qualifiers: The EXAMINE command analyzes expressions arithmetically. Therefore, qualifiers are interpreted correctly only when they appear immediately after the command name.

Qualifiers

/ASCII
Displays the data at the specified location in ASCII format.

Binary values that do not have ASCII equivalents are displayed as periods (.).

When you specify the /ASCII qualifier, or when ASCII mode is the default, hexadecimal is used as the default radix for numeric literals that are specified on the command line.

/BYTE
Displays data at the specified location 1 byte at a time.

/DECIMAL
Displays the contents of the specified location in decimal format.

/HEXADECIMAL
Displays the contents of the specified location in hexadecimal format.

/LONGWORD
Displays data at the specified location 1 longword at a time.

/OCTAL
Displays the contents of the specified location in octal format.

/WORD
Displays data at the specified location 1 word at a time.

Examples

1.
```
$ RUN   MYPROG
Ctrl/Y
$ EXAMINE   2678
0002678:  1F4C5026
$ CONTINUE
```

   In this example, the RUN command begins execution of the image MYPROG.EXE. While MYPROG is running, pressing Ctrl/Y interrupts its execution, and the EXAMINE command displays the contents of virtual memory location 2678 (hexadecimal).

2.
```
$ BASE = %X1C00
$ READBUF = BASE + %X50
$ ENDBUF = BASE + %XA0
$ RUN   TEST
Ctrl/Y
$ EXAMINE/ASCII READBUF:ENDBUF
00001C50:  BEGINNING OF FILE MAPPED TO GLOBAL SECTION
   .
   .
   .
```

   In this example, before executing the program TEST.EXE, symbolic names are defined for the program's base address and for labels READBUF and ENDBUF; all are expressed in hexadecimal format using the radix operator %X. READBUF and ENDBUF define offsets from the program base.

   While the program is executing, pressing Ctrl/Y interrupts it, and the EXAMINE command displays in ASCII format all data between the specified memory locations.

EXCHANGE

Invokes the Exchange utility (EXCHANGE), which manipulates mass storage volumes that are written in formats other than those normally recognized by the operating system.

EXCHANGE allows you to perform any of the following tasks:

- Create foreign volumes.
- Transfer files to and from the volume.
- List directories of the volume.

For block-addressable devices, such as RT–11 disks, EXCHANGE performs additional operations such as renaming and deleting files. EXCHANGE can also manipulate Files–11 files that are images of foreign volumes; these files are called **virtual devices**.

For a complete description of EXCHANGE, see the *OpenVMS Exchange Utility Manual* and the online help.

Format

EXCHANGE [subcommand] [filespec] [filespec]

EXCHANGE/NETWORK

Enables the operating system to transfer files to or from operating systems that do not support OpenVMS file organizations. The transfer occurs over a DECnet network communications link that connects OpenVMS systems and non OpenVMS operating system nodes.

Using DECnet services, the EXCHANGE/NETWORK command can perform any of the following tasks:

- Transfer files between an OpenVMS node and a non OpenVMS system node.
- Transfer a group of input files to a group of output files.
- Transfer files between two non OpenVMS nodes, provided those nodes share DECnet connections with the OpenVMS node that issues the EXCHANGE /NETWORK command.

Format

EXCHANGE/NETWORK input-filespec[,...] output-filespec

Parameters

input-filespec[,...]

Specifies the name of an existing file to be transferred. The asterisk (*) and the percent sign (%) wildcard characters are allowed. If you specify more than one file, separate the file specifications with commas (,).

output-filespec

Specifies the name of the output file into which the input is transferred.

You must specify at least one field in the output file specification. If you omit the device or directory, your current default device and directory are used. The EXCHANGE/NETWORK command replaces any other missing fields (file name, file type, and version number) with the corresponding field of the input file specification.

The EXCHANGE/NETWORK command creates a new output file for every input file that you specify.

You can use the asterisk (*) wildcard character in place of the file name, the file type, or the version number. The EXCHANGE/NETWORK command uses the corresponding field in the related input file to name the output file. You can also use the asterisk (*) wildcard character in the output file specification to direct EXCHANGE/NETWORK to create more than one output file. For example:

```
$  EXCHANGE/NETWORK A.A,B.B  MYPC::*.C
```

This EXCHANGE/NETWORK command creates the files A.C and B.C at the non OpenVMS target node MYPC.

A more complete explanation of the asterisk (*) and the percent sign (%) wildcard characters and version numbers follows in the Description section.

EXCHANGE/NETWORK

Description

The EXCHANGE/NETWORK command imposes the following restrictions:

- Transfers of files can occur only between disk devices. (If a disk device is not the desired permanent residence for the file, you must either move the file to a disk before issuing the command or retrieve the file from a disk after the command completes.)
- The remote system must have a block size of 512 bytes, where a byte is 8 bits long.
- The nodes transferring files must support the DECnet Data Access Protocol (DAP).

The OpenVMS Record Management Services (RMS) facility provides the operating system access to records in OpenVMS RMS files. To transfer OpenVMS RMS files between two nodes where both nodes are OpenVMS nodes, use one of the other DCL commands (such as COPY, APPEND, or CONVERT), as appropriate. These commands recognize RMS file organizations and are designed to ensure that RMS record structures are preserved as your files are moved.

Use the EXCHANGE/NETWORK command to transfer files between OpenVMS nodes and non OpenVMS nodes when the differences in the file organizations would otherwise prevent the transfer or could lead to undesirable results. While using the COPY command ensures that both the contents and the attributes of a replicated file are preserved, the EXCHANGE/NETWORK command has more advantages. The EXCHANGE/NETWORK command offers you explicit control of your record attributes during file transfers, with the opportunity to make a file usable on several different operating systems.

The EXCHANGE/NETWORK command transfers files between OpenVMS nodes and non OpenVMS nodes connected to the same DECnet network. If the non OpenVMS system does not support OpenVMS file organizations, the EXCHANGE /NETWORK command can modify or discard file and record attributes during the transfer. However, if the target system is an OpenVMS node, you have the option of applying new file and record attributes to the output file by supplying a File Definition Language (FDL) file, as described later in this section. The EXCHANGE/NETWORK command provides a number of defaults to handle the majority of transfers properly. However, in some situations, you need to know your file or record format requirements at both nodes.

OpenVMS File and Record Attributes

All RMS files in the OpenVMS environment include stored information, known as the file and record attributes, to describe the file and record characteristics. File attributes consist of items such as file organization, file protection, and file allocation information. Record attributes consist of items such as the record format, record size, key definitions for indexed files, and carriage control information. These attributes define the data format and access methods for the OpenVMS RMS facility.

NonOpenVMS operating systems that do not support OpenVMS file organizations have no means of storing file and record attributes with their files. Transferring an OpenVMS file to a non OpenVMS system that is unable to store and handle file and record attributes can result in most of this information being discarded. Removing these attributes from a file can render it useless if it must be returned to the OpenVMS system.

Transferring Files to OpenVMS Nodes

When you transfer files to an OpenVMS system from a non OpenVMS system, the files typically assume default file and record attributes. However, you can specify the attributes that you want the file to acquire in a File Definition Language (FDL) file. Alternatively, if transferring a CDA document, enter the following command after the EXCHANGE/NETWORK command:

```
$  SET FILE/SEMANTICS=[ddif,dtif] document-name.doc
```

If you specify an FDL file with the /FDL qualifier, the FDL file determines the characteristics of the output file. This feature is useful in establishing compatible file and record attributes when you transfer a file from a non OpenVMS system to an OpenVMS system. However, when you use an FDL file, you also assume responsibility for determining the required characteristics.

For more information on FDL files, see the *OpenVMS Record Management Utilities Reference Manual*.

Transferring Files to Non-OpenVMS Nodes

The EXCHANGE/NETWORK command discards file and record attributes associated with an OpenVMS file during a transfer to a non OpenVMS system that does not support OpenVMS file organizations. Be aware that the loss of file and record attributes in the transfer can render the output file useless for many applications.

Selecting Transfer Modes

The EXCHANGE/NETWORK command has four transfer mode options: AUTOMATIC, BLOCK, RECORD, and CONVERT. For most file transfers, AUTOMATIC is sufficient. The AUTOMATIC transfer mode option allows the EXCHANGE/NETWORK command to transfer files using either block or record I/O. The selection is based on the input file organization and the operating systems involved.

Selecting the BLOCK transfer mode option forces the EXCHANGE/NETWORK command to open both the input and output files for block I/O access. The input file is then transferred to the output file block by block. Use this transfer mode when you transfer executable images. It is also useful when you must preserve a file's content exactly, which is a common requirement when you store files temporarily on another system or when cooperating applications exist on the systems.

Selecting the RECORD transfer mode option forces the EXCHANGE/NETWORK command to open both the input file and output file for record I/O access. The input file is then transferred to the output file record by record. This transfer mode is primarily used for transferring text files.

Selecting the CONVERT transfer mode option forces the EXCHANGE/NETWORK command to open the input file for RECORD access and the output file for BLOCK access. Records are then read in from the input file, packed into blocks, and are written to the output file. This transfer mode is primarily used for transferring files with no implied carriage control. For example, to transfer a file created with DIGITAL Standard Runoff (DSR) to a DECnet DOS system, you must use the CONVERT transfer mode option. To transfer the resultant output file back to an OpenVMS node, use the AUTOMATIC transfer mode option.

Wildcard Characters

The asterisk (*) and the percent sign (%) wildcard characters are permitted in the file specifications and follow the behavior typical of other OpenVMS system commands with respect to the OpenVMS node.

When more than one input file is specified, but the asterisk (*) or the percent sign (%) wildcard characters are not specified in the output file specification, the first input file is copied to the output file, and each subsequent input file is transferred and given a higher version number of the same output file name. Note that the files are not concatenated into a single output file. Also note that when you transfer files to foreign systems that do not support version numbers, only one output file results, and it is the last input file.

To create multiple output files, specify multiple input files and use at least one of the following:

- An asterisk (*) wildcard character in the output file name, file type, or version number field
- Only a node name, a device name, or a directory specification as the output file specification

When you create multiple output files, the EXCHANGE/NETWORK command uses the corresponding field from each input file in the output file name.

Use the /LOG qualifier when you specify multiple input and output files to verify that the files were copied as you intended.

Version Numbers

The following guidelines apply when the target node file formats accept version numbers.

If no version numbers are specified for input and output files, the EXCHANGE /NETWORK command (by default) assigns a version number to the output files that is either of the following:

- The version number of the input file
- A version number one greater than the highest version number of an existing file with the same file name and file type

When the output file version number is specified by an asterisk (*) wildcard character, the EXCHANGE/NETWORK command uses the version numbers of the associated input files as the version numbers of the output files.

If the output file specification has an explicit version number, the EXCHANGE /NETWORK command normally uses that number for the output file specification. However, if an equal or higher version of the output file already exists, no warning message is issued, the file is copied, and the version number is set to a value one greater than the highest version number already existing.

File Protection and Creation/Revision Dates

The EXCHANGE/NETWORK command treats an output file as a new file when any portion of the output file name is specified explicitly. When the output node is an OpenVMS system, the creation date for a new file is set to the current time and date. However, if the output file specification consists *only* of the asterisk (*) and the percent sign (%) wildcard characters, the output file no longer qualifies as a new file, and, therefore, the creation date of the input file is used. That is, if the output file specification is one of the following, the creation date becomes that of the input file: *, *.*, or *.*;*.

The revision date of the output file is always set to the current time and date; the backup date is set to zero. The output file is assigned a new expiration date. (Expiration dates are set by the file system if retention is enabled; otherwise, they are set to zero.)

When the target node is an OpenVMS node, the protection and access control list (ACL) of the output file is determined by the following parameters, in the following order:

1. Protection of previously existing versions of the output file
2. Default protection and ACL of the output directory
3. Process default file protection

For an introduction to ACLs, see the *OpenVMS Guide to System Security*.

On OpenVMS systems, the owner of the output file usually is the same as the creator of the output file. However, if a user with extended privileges creates the output file, the owner is either the owner of the parent directory or the owner of a previous version of the output file, if one exists.

Extended privileges include any of the following:

- SYSPRV (system privilege) or BYPASS
- System user identification code (UIC)
- GRPPRV (group privilege) if the owner of the parent directory (or previous version of the output file) is in the same group as the creator of the new output file
- An identifier (with the resource attribute) representing the owner of the parent directory (or previous version of the output file)

Qualifiers

/BACKUP

Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /BACKUP qualifier selects files according to the dates of their most recent backups. This qualifier is incompatible with the /CREATED, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you do not specify any of these four time qualifiers, the default is the /CREATED qualifier.

/BEFORE[=time]

Selects only those files dated prior to the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following qualifiers with the /BEFORE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

/BY_OWNER[=uic]

Selects only those files whose owner user identification code (UIC) matches the specified owner UIC. The default UIC is that of the current process.

Specify the UIC by using standard UIC format as described in the *OpenVMS Guide to System Security*.

/CONFIRM
/NOCONFIRM (default)
Controls whether a request is issued before each file transfer operation to confirm that the operation should be performed on that file. The following responses are valid:

YES	NO	QUIT
TRUE	FALSE	Ctrl/Z
1	0	ALL
	Return	

You can use any combination of uppercase and lowercase letters for word responses. Word responses can be abbreviated to one or more letters (for example, T, TR, or TRU for TRUE), but these abbreviations must be unique. Affirmative answers are YES, TRUE, and 1. Negative answers include: NO, FALSE, 0, and pressing the Return key. Entering QUIT or pressing Ctrl/Z indicates that you want to stop processing the command at that point. When you respond by entering ALL, the command continues to process, but no further prompts are given. If you type a response other than one of those in the list, DCL issues an error message and redisplays the prompt.

/CREATED (default)
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /CREATED qualifier selects files based on their dates of creation. This qualifier is incompatible with the /BACKUP, /EXPIRED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you do not specify any of these four time qualifiers, the default is the /CREATED qualifier.

/EXCLUDE=(filespec[,...])
Excludes the specified files from the file transfer operation. You can include a directory but not a device in the file specification. The asterisk (*) and the percent sign (%) wildcard characters are allowed in the file specification. However, you cannot use relative version numbers to exclude a specific version. If you specify only one file, you can omit the parentheses.

/EXPIRED
Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /EXPIRED qualifier selects files according to their expiration dates. (The expiration date is set with the SET FILE/EXPIRATION_DATE command.) The /EXPIRED qualifier is incompatible with the /BACKUP, /CREATED, and /MODIFIED qualifiers, which also allow you to select files according to time attributes. If you do not specify any of these four time qualifiers, the default is the /CREATED qualifier.

/FDL=fdl-filespec
Specifies that the output file characteristics are described in the File Definition Language (FDL) file. Use this qualifier when you require special output file characteristics. For more information about FDL files, see the *OpenVMS Record Management Utilities Reference Manual*.

Use of the /FDL qualifier implies that the transfer mode is block by block. However, the transfer mode you specify with the /TRANSFER_MODE qualifier prevails.

/LOG
/NOLOG (default)

Controls whether the EXCHANGE/NETWORK command displays the file specifications of each file copied.

When you use the /LOG qualifier, the EXCHANGE/NETWORK command displays the following for each copy operation:

- The file specifications of the input and output files
- The number of blocks or the number of records copied (depending on whether the file is copied on a block-by-block or record-by-record basis)

/MODIFIED

Modifies the time value specified with the /BEFORE or the /SINCE qualifier. The /MODIFIED qualifier selects files according to the date on which they were last modified. This time qualifier is incompatible with the /BACKUP, /CREATED, and /EXPIRED qualifiers, which also allow you to select files according to time attributes. If you do not specify any of these four time qualifiers, the default is the /CREATED qualifier.

/SINCE[=time]

Selects only those files dated after the specified time. You can specify time as absolute time, as a combination of absolute and delta times, or as one of the following keywords: BOOT, LOGIN, TODAY (default), TOMORROW, or YESTERDAY. Specify one of the following time qualifiers with the /SINCE qualifier to indicate the time attribute to be used as the basis for selection: /BACKUP, /CREATED (default), /EXPIRED, or /MODIFIED.

For complete information about specifying time values, see the *OpenVMS User's Manual*.

/TRANSFER_MODE=option

Specifies the I/O method to be used in the transfer. This qualifier is useful for all file formats. You can specify any one of the following options:

Option	Function
AUTOMATIC	Allows the EXCHANGE/NETWORK command to determine the appropriate transfer mode. This is the default transfer mode.
BLOCK	Opens both the input and output files for block I/O and transfers the files block by block.
CONVERT[=option[,...]]	Reads records from the input file, packs them into blocks, and writes them to the output file in block mode. The options listed in the following table determine what additional information is inserted during the transfer.

Option	Function
RECORD	Opens both the input and output files for record I/O and transfers the files record by record. The target system must support record operations, and the input file must be record oriented.

The following four options are available with the CONVERT transfer mode to control the insertion of special characters in the records:

Option	Function
CARRIAGE_CONTROL	Any carriage control information in the input file is interpreted, is expanded into actual characters, and is included with each record.
COUNTED	The length of each record, in bytes, is included at the beginning of the record. The length includes all FIXED_CONTROL, CARRIAGE_CONTROL, and RECORD_ SEPARATOR information in each record.
FIXED_CONTROL	All variable length with fixed control record (VFC) information is written to the output file as part of the data. This information follows the record length information, if the COUNTED option was specified.
RECORD_SEPARATOR= *separator*	A 1- or 2-byte record separator is inserted between each record. Record separator characters are the last characters in the record. The three choices for separator characters are as follows: • CR: Specifies carriage return only. • LF: Specifies line feed only. • CRLF: Specifies carriage return and line feed.

Examples

1. `$ EXCHANGE/NETWORK VMS_FILE.DAT KUDOS::FOREIGN_SYS.DAT`

   In this example, the EXCHANGE/NETWORK command transfers the file VMS_FILE.DAT located in the current default device and directory to the file FOREIGN_SYS.DAT on the non OpenVMS node KUDOS. Because the /TRANSFER_MODE qualifier was not explicitly specified, the EXCHANGE /NETWORK command automatically determines whether the transfer method should be block or record I/O.

2.
```
$ EXCHANGE/NETWORK/TRANSFER_MODE=BLOCK -
_$ KUDOS::FOREIGN_SYS.DAT VMS_FILE.DAT
```

   In this example, the EXCHANGE/NETWORK command transfers the file FOREIGN_SYS.DAT from the non OpenVMS node KUDOS to the file VMS_FILE.DAT in the current default device and directory. Block I/O is specified for the transfer mode.

3.
```
$ EXCHANGE/NETWORK/FDL=VMS_FILE_DEFINITION.FDL -
_$ KUDOS::REMOTE_FILE.TXT  VMS_FILE.DAT
```

   In this example, the EXCHANGE/NETWORK command transfers the file REMOTE_FILE.TXT on node KUDOS to the file VMS_FILE.DAT. The file attributes for the output file VMS_FILE.DAT are obtained from the File Definition Language (FDL) source file VMS_FILE_DEFINITION.FDL. For more information about creating FDL files, see the *OpenVMS Record Management Utilities Reference Manual*. Because the qualifier /FDL is specified and the /TRANSFER_MODE qualifier is omitted, the transfer mode uses block I/O, by default.

4.
```
$ EXCHANGE/NETWORK -
_$ /TRANSFER_MODE=CONVERT=(CARRIAGE_CONTROL,COUNTED, -
_$ RECORD_SEPARATOR=CRLF,FIXED_CONTROL) -
_$ PRINT_FILE.TXT  KUDOS::*
```

   In this example, the EXCHANGE/NETWORK command transfers the file PRINT_FILE.TXT from the current default device and directory to the file PRINT_FILE.TXT on the non OpenVMS node KUDOS. The use of the CONVERT option with the /TRANSFER_MODE qualifier forces the input file to be read in record by record, modified as specified by the CONVERT options that follow, and written to the output file block by block. As many records as will fit are packed into the output blocks.

   The CONVERT option CARRIAGE_CONTROL specifies that carriage control information is converted to ASCII characters and is inserted before the data or is appended to the record, depending on whether prefix control or postfix control, or both, are used. The CONVERT option FIXED_CONTROL specifies that any fixed control information be translated to ASCII characters and be inserted at the beginning of the record. The CONVERT option RECORD_SEPARATOR=CRLF appends the two specified characters, carriage return and line feed, to the end of the record. The CONVERT option COUNTED specifies that the total length of the record must be counted (once the impact of all the previous convert options have been added), and the result is to be inserted at the beginning of the record, in the first 2 bytes.

EXIT

Terminates processing of a command procedure or subroutine and returns control to the calling command level—either an invoking command procedure or interactive DCL. The EXIT command also terminates an image normally after a user enters Ctrl/Y (executing another image has the same effect).

Format

EXIT [status-code]

Parameter

status-code

Defines a numeric value for the reserved global symbol $STATUS. You can specify the status-code parameter as an integer or an expression equivalent to an integer value. The value can be tested by the next outer command level. The low-order 3 bits of the value determine the value of the global symbol $SEVERITY.

If you specify a status code, DCL interprets the code as a condition code. Note that even numeric values produce warning, error, and fatal error messages, and that odd numeric values produce either no message or a success or informational message.

If you do not specify a status code, the current value of $STATUS is saved. When control returns to the outer command level, $STATUS contains the status of the most recently executed command or program.

Description

The EXIT and STOP commands both provide a way to terminate the execution of a procedure. The EXIT command terminates execution of the current command procedure and returns control to the calling command level. If you enter the EXIT command from a noninteractive process (such as a batch job), at command level 0, then the process terminates.

The STOP command returns control to command level 0, regardless of the current command level. If you execute the STOP command from a command procedure or from a noninteractive process (such as a batch job), the process terminates.

When a DCL command, user program, or command procedure completes execution, the command interpreter saves the condition code value in the global symbol $STATUS. If an EXIT command does not explicitly set a value for $STATUS, the command interpreter uses the current value of $STATUS to determine the error status.

The low-order 3 bits of the status value contained in $STATUS represent the severity of the condition. The reserved global symbol $SEVERITY contains this portion of the condition code. Severity values range from 0 to 4, as follows:

Value	Severity
0	Warning
1	Success
2	Error
3	Information
4	Severe (fatal) error

Note that the success and information codes have odd numeric values, and that warning and error codes have even numeric values.

When any command procedure exits and returns control to another level, the command interpreter tests the current value of $STATUS. If $STATUS contains an even numeric value and if its high-order bit is 0, the command interpreter displays the system message associated with that status code, if one exists. (If no message exists, the message NOMSG will be displayed.) If the high-order bit is 1, the message is not displayed.

When a command procedure exits following a warning or error condition that has already been displayed by a DCL command, the command interpreter sets the high-order bit of $STATUS to 1, leaving the remainder of the value intact. This ensures that error messages are not displayed by both the command that caused the error, and by the command procedure.

The EXIT command, when used after you interrupt an image with Ctrl/Y, causes a normal termination of the image that is currently executing. If the image declared any exit-handling routines, they are given control. This is in contrast to the STOP command, which does not execute exit-handling routines. For this reason, the EXIT command is generally preferable to the STOP command.

Examples

1.
```
$ EXIT 1
```

   The EXIT command in this example exits to the next higher command level, giving $STATUS and $SEVERITY a value of 1.

2.
```
$ ON WARNING THEN EXIT
$ FORTRAN 'P1'
$ LINK 'P1'
$ RUN 'P1'
```

   The EXIT command in this example is used as the target of an ON command; this statement ensures that the command procedure terminates whenever any warnings or errors are issued by any command in the procedure.

   The procedure exits with the status value of the command or program that caused the termination.

3.
```
$ START:
$        IF (P1 .EQS. "TAPE") .OR. (P1 .EQS. "DISK") THEN GOTO 'P1'
$        INQUIRE P1 "Enter device (TAPE or DISK)"
$        GOTO START
$ TAPE: !  Process tape files
   .
   .
   .
$        EXIT
$ DISK:  ! Process disk files
   .
   .
   .
$        EXIT
```

The command procedure in this example shows how to use the EXIT command to terminate different command paths within the procedure. To execute the procedure, you must enter either TAPE or DISK as a parameter. The IF command uses a logical OR to test whether either of these strings was entered. If the result is true, the GOTO command branches to the corresponding label. If P1 was neither TAPE nor DISK, the INQUIRE command prompts for a correct parameter.

The commands following each of the labels TAPE and DISK provide different paths through the procedure. The EXIT command before the label DISK ensures that the commands after the label DISK are executed only if the procedure explicitly branches to DISK.

Note that the EXIT command at the end of the procedure is not required because the end of the procedure causes an implicit EXIT command. Use of the EXIT command, however, is recommended.

4.
```
$ IF P1. EQS. "" THEN -
     INQUIRE P1 "Enter filespec (null to exit)"
$ IF P1 .EQS. "" THEN EXIT
$ PRINT 'P1'/AFTER=20:00/COPIES=50/FORMS=6
```

The command procedure in this example tests whether a parameter was passed to it; if the parameter was not passed, the procedure prompts for the required parameter. Then it retests the parameter P1. If a null string, indicated by a carriage return for a line with no data, is entered, the procedure exits. Otherwise, it executes the PRINT command with the current value of P1 as the input parameter.

5.
```
$ IF P1 .EQS. "" THEN INQUIRE P1 "Code"
$ CODE = %X'P1'
$ EXIT CODE
```

The command procedure in this example, E.COM, illustrates how to determine the system message, if any, associated with a hexadecimal system status code. The procedure requires a parameter and prompts if none is entered. Then it prefixes the value with the radix operator %X and assigns this string to the symbol CODE. Finally, it issues the EXIT command with the hexadecimal value. The following example uses the procedure E.COM:

```
$ @E 1C
%SYSTEM-F-EXQUOTA, exceeded quota
```

When the procedure exits, the value of $STATUS is %X1C, which equates to the EXQUOTA message. Note that you can also use the F$MESSAGE lexical function to determine the message that corresponds to a status code.

6.
```
$ RUN MYPROG
Ctrl/Y
$ EXIT
```

In this interactive example, the RUN command initiates execution of the image MYPROG.EXE. Then pressing Ctrl/Y interrupts the execution. The EXIT command that follows calls any exit handlers declared by the image before terminating MYPROG.EXE.

FONT

Converts an ASCII bitmap distribution format (BDF) into binary portable compiled format (PCF) on Alpha systems and into binary server natural form (SNF) on VAX systems. The DECwindows server uses a PCF or SNF file to display a font. In addition to converting the BDF file to binary form, the font compiler provides statistical information about the font and the compilation process. For more information about using the font compiler, refer to the OpenVMS DECwindows programming documentation.

Format

FONT filespec

GOSUB

Transfers control to a labeled subroutine in a command procedure without creating a new procedure level.

Format

GOSUB label

Parameter

label
Specifies a label of 1 to 255 alphanumeric characters that appears as the first item on a command line. A label may not contain embedded blanks. When the GOSUB command is executed, control passes to the command following the specified label.

The label can precede or follow the GOSUB statement in the current command procedure. When you use a label in a command procedure, it must be terminated with a colon (:). If you use duplicate labels, control is always given to the label most recently read by DCL.

Description

Use the GOSUB command in command procedures to transfer control to a subroutine specified by the label. If the command stream is not being read from a random-access device (that is, a disk device), the GOSUB command performs no operation.

The RETURN command terminates the GOSUB subroutine procedure, returning control to the command following the calling GOSUB statement. The RETURN command accepts an optional status value.

The GOSUB command does not cause the creation of a new procedure level. Therefore, it is referred to as a ''local'' subroutine call. Any labels and local symbols defined in the current command procedure level are available to a subroutine invoked with a GOSUB command. The GOSUB command can be nested up to a maximum of 16 levels per procedure level.

When the command interpreter encounters a label, it enters the label in a label table. This table is allocated from space available in the local symbol table. If the command interpreter encounters a label that already exists in the table, the new definition replaces the existing one. Therefore, if you use duplicate labels, control is always given to the label most recently read by DCL. The following rules apply:

- If duplicate labels precede and follow the GOSUB command, control is given to the label preceding the command.
- If duplicate labels all precede the GOSUB command, control is given to the most recent label, that is, the one nearest the GOSUB command.
- If duplicate labels all follow the GOSUB command, control is given to the one nearest the GOSUB command.

If a label does not exist in the current command procedure, the procedure cannot continue and is forced to exit.

Note that the amount of space available for labels is limited. If a command procedure uses many symbols and contains many labels, the command interpreter may run out of table space and issue an error message.

Example

```
$!
$! GOSUB.COM
$!
$ SHOW TIME
$ GOSUB TEST1
$ WRITE SYS$OUTPUT "success completion"
$ EXIT
$!
$! TEST1 GOSUB definition
$!
$ TEST1:
$     WRITE SYS$OUTPUT "This is GOSUB level 1."
$     GOSUB TEST2
$     RETURN %X1
$!
$! TEST2 GOSUB definition
$!
$ TEST2:
$     WRITE SYS$OUTPUT "This is GOSUB level 2."
$     GOSUB TEST3
$     RETURN
$!
$! TEST3 GOSUB definition
$!
$ TEST3:
$     WRITE SYS$OUTPUT "This is GOSUB level 3."
$     RETURN
```

This sample command procedure shows how to use the GOSUB command to transfer control to labeled subroutines. The GOSUB command transfers control to the subroutine labeled TEST1. The procedure executes the commands in subroutine TEST1, branching to the subroutine labeled TEST2. The procedure then executes the commands in subroutine TEST2, branching to the subroutine labeled TEST3. Each subroutine is terminated by the RETURN command. After TEST3 is executed, the RETURN command returns control back to the command line following each calling GOSUB statement. At this point, the procedure has been successfully executed.

GOTO

Transfers control to a labeled statement in a command procedure.

Format

GOTO label

Parameter

label
Specifies a label of 1 to 255 alphanumeric characters that appears as the first item on a command line. A label cannot contain embedded blanks. When the GOTO command is executed, control passes to the command following the specified label.

When you use a label in a command procedure, it must be terminated with a colon (:). If you use duplicate labels, control is always given to the label most recently read by DCL.

Description

Use the GOTO command in command procedures to transfer control to a line that is not the next line in the procedure. The label can precede or follow the GOTO statement in the current command procedure. If the command stream is not being read from a random-access device (that is, a disk device), the GOTO command performs no operation.

If the target label of a GOTO command is inside a separate IF-THEN-ELSE construct, an error message (DCL-W-USGOTO) is returned.

When the command interpreter encounters a label, it enters the label in a label table. This table is allocated from space available in the local symbol table. If the command interpreter encounters a label that already exists in the table, the new definition replaces the existing one. Therefore, if you use duplicate labels, control is always given to the label most recently read by DCL. In general:

- If duplicate labels precede and follow the GOTO command, control is given to the label preceding the command.
- If duplicate labels all precede the GOTO command, control is given to the most recent label, that is, the one nearest the GOTO command.
- If duplicate labels all follow the GOTO command, control is given to the one nearest the GOTO command.

If a label does not exist in the current command procedure, the procedure cannot continue and is forced to exit.

Note that the amount of space available for labels is limited. If a command procedure uses many symbols and contains many labels, the command interpreter may run out of table space and issue an error message.

Examples

1.
```
$ IF P1 .EQS. "HELP" THEN GOTO TELL
$ IF P1 .EQS. "" THEN GOTO TELL
   .
   .
   .
$ EXIT
$ TELL:
$ TYPE SYS$INPUT
To use this procedure, you must enter a value for P1.
   .
   .
   .
$ EXIT
```

In this example, the IF command checks the first parameter passed to the command procedure; if this parameter is the string HELP or if the parameter is not specified, the GOTO command is executed and control is passed to the line labeled TELL. Otherwise, the procedure continues executing until the EXIT command is encountered. At the label TELL, a TYPE command displays data in the input stream that documents how to use the procedure.

2.
```
$ ON ERROR THEN GOTO CHECK
   .
   .
   .
$ EXIT
$ CHECK:  ! Error handling routine
   .
   .
   .
$ END:
$ EXIT
```

The ON command establishes an error-handling routine. If any command or procedure subsequently executed in the command procedure returns an error or severe error, the GOTO command transfers control to the label CHECK.

HELP

The HELP command invokes the HELP Facility to display information about use of the system, including formats and explanations of commands, parameters, qualifiers and system messages. In response to the Topic? prompt, you can:

- Type the name of the command or topic for which you need help.
- Type INSTRUCTIONS for more detailed instructions on how to use HELP.
- Type HINTS if you are not sure of the name of the command or topic for which you need help.
- Type /MESSAGE for help with the HELP/MESSAGE utility.
- Type a question mark (?) to redisplay the most recently requested text.
- Press the RETURN key one or more times to exit from HELP.

You can abbreviate any topic name, although ambiguous abbreviations result in all matches being displayed.

Format

HELP [topic[subtopic...]]

Parameter

topic[subtopic...]
Specifies the topics or topic and subtopics on which you want information from a help library.

Description

Information within help libraries is arranged in a hierarchical manner. The levels are as follows:

1. None—If you do not specify a keyword, the Help facility describes the HELP command and lists the topics that are documented in the root library. Each item in the list is a keyword in the first level of the hierarchy.
2. Topic-name—If you specify a keyword by naming a topic, the Help facility describes the topic as it is documented in either the root library or in one of the other enabled default libraries. Keywords for additional information available on this topic are listed.
3. Topic-name subtopic—If you specify a subtopic following a topic, the Help facility provides a description of the specified subtopic.
4. @filespec followed by any of the previous levels—If you specify a help library to replace the current root library, the Help facility searches that library for a description of the topic or subtopic specified. The file specification must take the same form as the file specification included with the /LIBRARY command qualifier. However, if the specified library is an enabled user-defined default library, the file specification can be abbreviated to any unique leading substring of that default library's logical name translation.

HELP

To use the Help facility on OpenVMS in its simplest form, enter the HELP command from your terminal. The Help facility displays a list of topics at your terminal and the prompt Topic?. To see information on one of the topics, type the topic name after the prompt. The system displays information on that topic.

If the topic has subtopics, the HELP command lists the subtopics and displays the Subtopic? prompt. To get information on one of the subtopics, type the name after the prompt. To see information on another topic, press the Return key. You can now ask for information on another topic when the Help facility displays the Topic? prompt. Press the Return key to exit the Help facility and return to DCL command level.

If you use an asterisk (*) in place of any keyword, the HELP command displays all information available at the level that the asterisk replaces. For example, HELP COPY * displays all the subtopics under the topic COPY.

If you use an ellipsis (. . .) immediately after any primary keyword, the Help facility displays all the information on the specified topic and all subtopics of that topic. For example, HELP COPY . . . displays information on the COPY topic as well as information on all the subtopics under COPY. The ellipsis can only be used from the topic level; it cannot be used from the subtopic level.

The asterisk (*) and the percent sign (%) wildcard characters are allowed in the keyword.

Qualifiers

/EXACT
Use with the /PAGE=SAVE and /SEARCH qualifiers to specify a search string that must match the search string exactly and must be enclosed with quotation marks (‘‘ ’’).

If you specify the /EXACT qualifier without the /SEARCH qualifier, exact search mode is enabled when you set the search string with the Find (E1) key.

/HIGHLIGHT[=keyword]
/NOHIGHLIGHT (default)
Use with the /PAGE=SAVE and /SEARCH qualifiers to specify the type of highlighting you want when a search string is found. When a string is found, the entire line is highlighted. You can use the following keywords: BOLD, BLINK, REVERSE, and UNDERLINE. BOLD is the default highlighting.

/INSTRUCTIONS (default)
/NOINSTRUCTIONS
Displays an explanation of the HELP command along with the list of topics (if no topic is specified). By default, the HELP command display includes a description of the facility and the format, along with the list of topics. If you specify the /NOINSTRUCTIONS qualifier, only the list of topics is displayed.

/LIBLIST (default)
/NOLIBLIST
Displays any auxiliary help libraries.

/LIBRARY=filespec
/NOLIBRARY
Uses an alternate help library instead of the default system library, SYS$HELP:HELPLIB.HLB. The specified library is used as the main (root)

help library, and is searched for Help facility information before any user-defined default help libraries are checked.

If you omit the device and directory specification, the default is SYS$HELP, the logical name of the location of the system help libraries. The default file type is .HLB.

The /NOLIBRARY qualifier excludes the default help library from the library search order.

/MESSAGE
Displays descriptions of system messages. See the HELP/MESSAGE command in this manual.

/OUTPUT[=filespec]
/NOOUTPUT
Controls where the output of the command is sent. By default, the output is sent to SYS$OUTPUT, the current process default output stream or device.

If you enter the /OUTPUT qualifier with a partial file specification (for example, /OUTPUT=[JONES]), HELP is the default file name and LIS is the default file type. The asterisk (*) and the percent sign (%) wildcard characters are not allowed.

If you enter the /NOOUTPUT qualifier, output is suppressed.

/PAGE[=keyword]
/NOPAGE (default)
Controls the display of information on the screen.

You can use the following keywords with the /PAGE qualifier:

CLEAR_SCREEN	Clears the screen before each page is displayed.
SCROLL	Displays information one line at a time.
SAVE[=*n*]	Enables screen navigation of information, where *n* is the number of pages to store.

The /PAGE=SAVE qualifier allows you to navigate through screens of information. The /PAGE=SAVE qualifier stores up to 5 screens of up to 255 columns of information. When you use the /PAGE=SAVE qualifier, you can use the following keys to navigate through the information:

Key Sequence	Description
Up arrow (↑), Ctrl/B	Scroll up one line.
Down arrow (↓)	Scroll down one line.
Left arrow (←)	Scroll left one column.
Right arrow (→)	Scroll right one column.
Find (E1)	Specify a string to find when the information is displayed.
Insert Here (E2)	Scroll right one half screen.
Remove (E3)	Scroll left one half screen.
Select (E4)	Toggle 80/132 column mode.
Prev Screen (E5)	Get the previous page of information.

Key Sequence	Description
Next Screen (E6), Return, Enter, Space	Get the next page of information.
F10, Ctrl/Z	Exit. (Some utilities define these differently.)
Help (F15)	Display utility help text.
Do (F16)	Toggle the display to oldest/newest page.
Ctrl/W	Refresh the display.

The /PAGE qualifier is not compatible with the /OUTPUT qualifier.

/PROMPT (default)
/NOPROMPT

Permits you to solicit further information interactively. If you specify the /NOPROMPT qualifier, the Help facility returns you to DCL command level after it displays the requested information.

If the /PROMPT qualifier is in effect, one of four different prompts is displayed, requesting you to specify a particular help topic or subtopic. Each prompt represents a different level in the hierarchy of help information. The four prompt levels are as follows:

1. Topic?—The root library is the main library and you are not currently examining the Help facility information for a particular topic.
2. [library-spec] Topic?—The root library is a library other than the main library and you are not currently examining the Help facility information for a particular topic.
3. [keyword] Subtopic?—The root library is the main library and you are currently examining the Help facility information for a particular topic (or subtopic).
4. A combination of 2 and 3.

When you encounter one of these prompts, you can enter any one of the responses described in the following table:

Response	Current Prompt Environment	Action
keyword[...]	1,2	Searches all enabled libraries for the keyword.
	3,4	Searches additional help libraries for the current topic (or subtopic) for the keyword.
@filespec keyword[...]	1,2	Same as above, except that the library specified by @filespec is now the root library. If the specified library does not exist, the Help facility treats @filespec as a normal keyword.
		Displays a list of topics available in the root library.

Response	Current Prompt Environment	Action
	3,4	Same as above; treats @filespec as a normal keyword.
		Displays the list of subtopics of the current topic (or subtopics) for which help exists.
Return	1	Exits from the Help facility.
	2	Changes root library to main library.
	3,4	Prompts for a topic or subtopic at the next higher level.
Ctrl/Z	1,2,3,4	Exits from the Help facility.

/SEARCH="string"
Use with the /PAGE=SAVE qualifier to specify a string that you want to find in the information being displayed. Quotation marks are required for the /SEARCH qualifier, if you include spaces in the text string.

You can also dynamically change the search string by pressing the Find key (E1) while the information is being displayed. Quotation marks are not required for a dynamic search.

/USERLIBRARY=(level[,...])
/NOUSERLIBRARY
Names the levels of search for information in auxiliary libraries. The levels are as follows:

PROCESS	Libraries defined at process level
GROUP	Libraries defined at group level
SYSTEM	Libraries defined at system level
ALL	All libraries (default)
NONE	No libraries (same as the /NOUSERLIBRARY qualifier)

Auxiliary help libraries are libraries defined with the logical names HLP$LIBRARY, HLP$LIBRARY_1, HLP$LIBRARY_2, and so on. Libraries are searched for information in this order: root (current) library, main library (if not current), libraries defined at process level, libraries defined at group level, libraries defined at system level, and the root library. If the search fails, the root library is searched a second time so that the context is returned to the root library from which the search was initiated. The default is the /USERLIBRARY=ALL qualifier. If you specify only one level for the Help facility to search, you can omit the parentheses.

/WRAP
/NOWRAP (default)
Use with the /PAGE=SAVE qualifier to limit the number of columns to the width of the screen and to wrap lines that extend beyond the width of the screen to the next line.

The /NOWRAP qualifier extends lines beyond the width of the screen and can be seen when you use the scrolling (left and right) features provided by the /PAGE=SAVE qualifier.

Examples

1.
```
$ HELP
HELP
  .
  .  (HELP message text and list of topics)
  .
Topic?
```

In this example, the HELP command is entered without any qualifiers or parameters. This example produces a display of the help topics available from the root help library, SYS$HELP:HELPLIB.HLB.

If you enter one of the listed topics in response to the Topic? prompt, the Help facility displays information about that topic and a list of subtopics (if there are any). If one or more subtopics exist, the Help facility prompts you for a subtopic, as follows:

```
Topic? ASSIGN
ASSIGN
  .
  .  (HELP message text and subtopics)
  .
ASSIGN Subtopic?
```

If you type a subtopic name, the Help facility displays information about that subtopic, as follows:

```
ASSIGN Subtopic? Name
ASSIGN
  Name
  .
  .  (HELP message text and subtopics, if any)
  .
ASSIGN Subtopic?
```

If one or more sub-subtopics exist, the Help facility prompts you for a sub-subtopic; otherwise, as in the previous example, the facility prompts you for another subtopic of the topic you are currently inspecting.

Entering a question mark (?) redisplays the Help facility message and options at your current level. Pressing the Return key does either of the following:

- Moves you back to the previous help level if you are in a subtopic level.
- Terminates the Help facility if you are at the first level.

Pressing Ctrl/Z terminates the Help facility at any level.

2.
```
$ HELP COPY...
```

The HELP command in this example displays a description of the COPY command and of the command's parameters and qualifiers. Note that the ellipsis can be used only from the topic level; it cannot be used from the subtopic level.

3.
```
$ HELP/NOPROMPT ASSIGN/GROUP
  .
  .  (ASSIGN/GROUP HELP message)
  .
$
$ HELP/NOPROMPT/PAGE EDIT *
  .
  .  (HELP messages on all first-level EDIT subtopics)
  .
$
```

The two HELP commands request help on specific topics. In each case, the HELP command displays the help message you request and then returns you to DCL command level and the dollar sign prompt ($).

The first command requests help on the /GROUP qualifier of the ASSIGN command. The asterisk (*) in the second example is a wildcard character. It signals the Help facility to display information about all EDIT subtopics, which are then displayed in alphabetical order. The /NOPROMPT qualifier suppresses prompting in both sample commands. The /PAGE qualifier on the second HELP command causes output to the screen to stop after each screen of information is displayed.

4.
```
$ HELP FILL
Sorry, no documentation on FILL
Additional information available:
 .
 .  (list of first-level topics )
 .
Topic? @EDTHELP FILL
FILL
 .
 .  (FILL HELP message)
 .
@EDTHELP Topic?
```

When you enter a request for help on a topic that is not in the default help library, you can instruct the Help facility to search another help library for the topic. In this example, entering the command @EDTHELP FILL instructs the Help facility to search the help library SYS$HELP:EDTHELP.HLB for information on FILL, an EDT editor command. The Help facility displays the message and prompts you for another EDT editor topic.

5.
```
$ SET DEFAULT SYS$HELP
$ DEFINE HLP$LIBRARY EDTHELP
$ DEFINE HLP$LIBRARY_1 MAILHELP
$ DEFINE HLP$LIBRARY_2 BASIC
$ DEFINE HLP$LIBRARY_3 DISK2:[MALCOLM]FLIP
$ HELP REM
```

You can use logical names to define libraries for the Help facility to search automatically if it does not find the specified topic in the OpenVMS root help library. This sequence of commands instructs the Help facility to search libraries in addition to the default root library, SYS$HELP:HELPLIB.HLB.

The four DEFINE statements create logical names for the four user-defined help libraries that the Help facility is to search after it has searched the root library. The first three entries are help libraries in the current default directory. By default, the Help facility searches for user-defined help libraries in the directory defined by the logical name SYS$HELP. The fourth entry is the help library FLIP.HLB in the directory DISK2:[MALCOLM]. Note that the

logical names that you use to define these help libraries must be numbered consecutively; that is, you cannot skip any numbers.

The Help facility first searches the root library for REM. It then searches the libraries HLP$LIBRARY, HLP$LIBRARY_1, HLP$LIBRARY_2, and so on, until it finds REM or exhausts the libraries it knows it can search. When it finds REM in the BASIC.HLB library, the Help facility displays the appropriate help information and prompts you for a subtopic in that library. If you request information on a topic not in the BASIC.HLB library, the Help facility once again searches the help libraries you have defined.

HELP/MESSAGE

Displays descriptions of system messages.

Format

HELP/MESSAGE [/qualifier [...]] [search-string]

Parameter

search-string
Specifies a message identifier or one or more words from a message's text. By default, HELP/MESSAGE displays a description of the message produced by the last executed command (that is, the message corresponding to the value currently stored in the CLI symbol $STATUS).

The Help Message utility (MSGHLP) operates on the search string using the following conventions:

- Words containing fewer than three alphanumeric characters are ignored.
- Words can be listed in any order.

  You can minimize search time by specifying the most unusual word first.
- Nonalphanumeric characters are ignored in the search. Exceptions are the percent sign (%) and hyphen (-) when they prefix a message. Therefore, you can paste a full message into the search string, provided you include these special characters and delete any variables (such as file names) that were inserted into the message.

  If Help Message fails to find a pasted message in the database, submit the command again and omit the leading special character, facility, and severity. Some common messages are documented as "shared" messages rather than facility-specific messages.
- Help Message matches all words that begin with the characters specified in the search string. Use /WORD_MATCH=WHOLE_WORD to specify that only whole words be matched.

Description

The Help Message utility accesses message descriptions in a text file. This text file is derived from the latest OpenVMS system messages documentation, and, optionally, from other source files, including user-supplied message documentation. By default, Help Message provides information on how the last executed command completed.

You can extract all messages produced by one or more specified facilities. By directing this output to a file, you can create and print your own customized message documentation.

For full details about adding comments or messages to the Help Message database, refer to the *OpenVMS System Messages: Companion Guide for Help Message Users*.

Qualifiers

/BRIEF
Outputs the message text only.

/DELETE=filename.MSGHLP
Deletes all messages contained in the specified .MSGHLP file from whichever of the following files is found first:

- A .MSGHLP$DATA file specified with the /LIBRARY qualifier
- The first .MSGHLP$DATA file in a search path specified by the /LIBRARY qualifier
- The first .MSGHLP$DATA file in the default search path (defined by logical name MSGHLP$LIBRARY)
- SYS$HELP:MSGHLP$LIBRARY.MSGHLP$DATA (the default .MSGHLP$DATA file)

You must have write access to Digital-supplied .MSGHLP$DATA files to delete messages from the Digital-supplied database.

Note

If you create a .MSGHLP file by specifying a search string, check the output .MSGHLP file to be sure the search did not pick up any unexpected messages that you do not want to delete from the database. Edit any such messages out of the .MSGHLP file before you perform the delete operation.

/EXTRACT=filename.MSGHLP
Extracts messages from the database and generates a .MSGHLP file that can be edited, if desired, and used as input for /INSERT and /DELETE operations. /EXTRACT retrieves data from a .MSGHLP$DATA file or logical search path specified by /LIBRARY or, by default, from files in the search path defined by logical name MSGHLP$LIBRARY. When /EXTRACT is not specified, Help Message produces output in standard text format by default (see /OUTPUT).

/FACILITY=?
/FACILITY=(facility-name [,...])
/FACILITY=ALL
Specifies which facilities in the database are to be searched for a match.

Enter /FACILITY=? to output a list of all facilities in the default database or in a database specified by /LIBRARY.

To narrow your search, specify one or more facility names with /FACILITY. (Multiple facilities must be enclosed in parentheses and be separated by commas.) Help Message then outputs only matching messages produced by the specified facility or facilities.

Specify /FACILITY=ALL to output messages for all facilities in the database. /FACILITY=ALL is the default unless another facility is implied; for example, specifying /STATUS or defaulting to the value of the CLI symbol $STATUS automatically identifies a specific facility. Similarly, cutting and pasting a message that includes a facility name invalidates use of the /FACILITY qualifier.

See the *OpenVMS System Messages: Companion Guide for Help Message Users* for more details about using the /FACILITY qualifier.

/FULL (default)
Outputs the complete message description, including message text, facility name, explanation, user action, and user-supplied comment, if any.

/INSERT=filename.MSGHLP
/INSERT=TT:
Updates the first of the following files to be found with new or changed information from the specified .MSGHLP file, or, if TT: is specified, with the data entered immediately at the terminal:

- A .MSGHLP$DATA file specified with the /LIBRARY qualifier
- The first .MSGHLP$DATA file in a search path specified by the /LIBRARY qualifier
- The first .MSGHLP$DATA file in the default search path (defined by logical logical name MSGHLP$LIBRARY)
- SYS$HELP:MSGHLP$LIBRARY.MSGHLP$DATA (the default .MSGHLP$DATA file)

You must have write access for the Digital-supplied .MSGHLP$DATA files to insert data into these files. User-supplied data is identified by change bars in Help Message output.

/LIBRARY=disk:[directory]filename.MSGHLP$DATA
/LIBRARY=disk:[directory]
/LIBRARY=logical-name
Defines the messages database for the current command to be a particular .MSGHLP$DATA file, all the .MSGHLP$DATA files in a specified directory, or all the files in a search path defined by a logical name.

For most operations, the default database is either SYS$HELP:MSGHLP$LIBRARY.MSGHLP$DATA or a search path of .MSGHLP$DATA files defined by the logical name MSGHLP$LIBRARY.

For /DELETE and /INSERT operations, the default database is either SYS$HELP:MSGHLP$LIBRARY.MSGHLP$DATA or the first file in a search path defined by the logical name MSGHLP$LIBRARY.

/OUTPUT=filespec
Writes output to the specified file. By default, Help Message writes output to SYS$OUTPUT, which is normally the terminal. (Use of /OUTPUT=filespec is incompatible with /PAGE.)

/PAGE (default for screen display)
/NOPAGE
Displays terminal output one screen at a time. The page length is automatically set to 1 line less than the value specified by SET TERMINAL/PAGE. (Use of /PAGE is incompatible with /OUTPUT=filespec.)

/SECTION_FILE=*
/SECTION_FILE=file-spec
Identifies the specified message section file to the system so that Help Message can interpret the $STATUS values for the messages in that file. The default file specification is SYS$MESSAGE:.EXE. Specifying /SECTION_FILE=*

automatically includes all OpenVMS-supplied message section files. For more information, see *OpenVMS System Messages: Companion Guide for Help Message Users*.

/SORT
/NOSORT (default)
Sorts output in alphabetical order. If a sort fails, retry the operation using the /WORK_FILES qualifier.

/STATUS=status-code
/STATUS='symbol'
/STATUS='$STATUS' (default)
Outputs the message corresponding to the specified status code. You can specify the status code with a decimal or hexadecimal number or a symbol enclosed in apostrophes. You can omit leading zeros, but you must prefix any hexadecimal number with "%X".

If a HELP/MESSAGE command does not include a search string, Help Message by default outputs the message corresponding to the CLI symbol $STATUS; that is, Help Message displays information on how the last executed command completed.

You cannot specify a search string or /FACILITY with /STATUS. /FACILITY is also illegal if you omit the search string and default to /STATUS='$STATUS'.

/WORD_MATCH=INITIAL_SUBSTRING (default)
/WORD_MATCH=WHOLE_WORD
/WORD_MATCH=INITIAL_SUBSTRING matches all words that begin with a word specified in the search string. The search string can contain multiple words to be matched. Only messages that match every word in the search string (in any order) are output.

/WORD_MATCH=WHOLE_WORD matches whole words only and refines your search to the exact words specified. For example, an exact search on ACC screens out dozens of other messages containing words that begin with the letters ACC.

/WORK_FILES=nn
/WORK_FILES=0 (default if qualifier is omitted)
/WORK_FILES=2 (default if qualifier is entered with no value)
Specifies that work files are to be used if the /SORT qualifier is specified. You can specify a value from 0 to 10 for *nn*. This qualifier has no effect if /SORT is not specified.

Examples

1.
```
$ SHOW DEVICE KUDOS
%SYSTEM-W-NOSUCHDEV, no such device available
$ HELP/MESSAGE
```

   The first command creates an error. The default HELP/MESSAGE command (with no qualifiers) displays a description of the SYSTEM facility message NOSUCHDEV.

2.
```
$ HELP/MESSAGE ACCVIO
$ HELP/MESSAGE/BRIEF ACCVIO
$ HELP/MESSAGE/FACILITY=SYSTEM ACCVIO
$ HELP/MESSAGE VIRTUAL ACCESS
$ HELP/MESSAGE/STATUS=12
$ HELP/MESSAGE/STATUS=%XC
```

These commands demonstrate how you can use various qualifiers to access and display the ACCVIO message (sometimes several!) in different formats.

3.
```
$ HELP/MESSAGE/BRIEF ACC
$ HELP/MESSAGE/BRIEF/WORD_MATCH=WHOLE_WORD ACC
```

In the first command, Help Message by default matches dozens of words beginning with the string ''ACC.'' The /WORD_MATCH=WHOLE_WORD qualifier dramatically refines the search to match the exact word only.

4.
```
$ HELP/MESSAGE/FACILITY=(BACKUP,SHARED)/SORT/OUTPUT=MESSAGES.TXT
```

This command selects all messages issued by the BACKUP facility and those messages documented as ''Shared by several facilities,'' alphabetizes them, and outputs them to a printable file called MESSAGES.TXT.

By selecting the messages you want and directing them to a file, you can create and print your own customized messages documentation.

5.
```
$ HELP/MESSAGE/EXTRACT=BADMESSAGE.MSGHLP BADMESSAGE
$ HELP/MESSAGE/DELETE=BADMESSAGE.MSGHLP-
_$ /LIBRARY=SYS$LOGIN:MYMESSAGES.MSGHLP$DATA
$ CONVERT SYS$LOGIN:MYMESSAGES.MSGHLP$DATA-
_$ SYS$LOGIN:MYMESSAGES.MSGHLP$DATA
$ PURGE SYS$LOGIN:MYMESSAGES.MSGHLP$DATA
$ HELP/MESSAGE/INSERT=BADMESSAGE.MSGHLP
```

The first command in this sequence extracts the hypothetical message BADMESSAGE from the default database and outputs it to file BADMESSAGE.MSGHLP.

The second command uses the BADMESSAGE.MSGHLP file to delete the BADMESSAGE description from the MYMESSAGES.MSGHLP$DATA file specified by the /LIBRARY qualifier.

The next two commands compress the MYMESSAGES.MSGHLP$DATA file to save disk space after the deletion.

The last command uses the BADMESSAGE.MSGHLP file (possibly an edited version at a later time) to insert the BADMESSAGE message into the default .MSGHLP$DATA file.

6.
```
$ HELP/MESSAGE/EXTRACT=NOSNO.MSGHLP NOSNO
$ EDIT/EDT NOSNO.MSGHLP
1NOSNO, can't ski; no snow
2XCSKI, XCSKI Program
3Your attempt to ski failed because there is no snow.
4Wait until there is snow and attempt the operation again.
5If you don't want to wait, go to a location where there is
5snow and ski there.
5
5Or, try ice skating instead!
[EXIT]
$ HELP/MESSAGE/INSERT=NOSNO.MSGHLP
```

This command sequence shows how users with write access to Digital-supplied .MSGHLP$DATA files can add a comment to a Digital-supplied message.

The first command extracts hypothetical message NOSNO to file NOSNO.MSGHLP. The second command edits the .MSGHLP file to add a comment at the end of the message. Each comment line, even blank lines, includes a ‘‘5’’ prefix. The next command updates the database by using NOSNO.MSGHLP to insert the updated message into the default .MSGHLP$DATA file.

IF

Tests the value of an expression and, depending on the syntax specified, executes the following:

- One command following the THEN keyword if the expression is true
- Multiple commands following the $THEN command if the expression is true
- One or more commands following the $ELSE command if the expression is false

Format

```
$ IF   expression THEN [$] command
```

or

```
$ IF   expression
$ THEN   [command]
command
    .
    .
    .

$ [ELSE]   [command]
command
    .
    .
    .

$ ENDIF
```

Note

Digital advises against assigning a symbolic name that is already a DCL command name. Digital especially discourages the assignment of symbols such as IF, THEN, ELSE, and GOTO, which can affect the interpretation of command procedures.

Parameters

expression

Defines the test to be performed. The expression can consist of one or more numeric constants, string literals, symbolic names, or lexical functions separated by logical, arithmetic, or string operators.

Expressions in IF commands are automatically evaluated during the execution of the command. Character strings beginning with alphabetic characters that are not enclosed in quotation marks (‘‘ ’’) are assumed to be symbol names or lexical functions. The command language interpreter (CLI) replaces these strings with their current values.

Symbol substitution in expressions in IF commands is not iterative; that is, each symbol is replaced only once. However, if you want iterative substitution, precede a symbol name with an apostrophe (') or ampersand (&).

The command interpreter does not execute an IF command when it contains an undefined symbol. Instead, the command interpreter issues a warning message and executes the next command in the procedure.

For a summary of operators and details on how to specify expressions, see the *OpenVMS User's Manual*.

command

Specifies the DCL command or commands to be executed, depending on the syntax specified, when the result of the expression is true or false.

Description

The IF command tests the value of an expression and executes a given command if the result of the expression is true. The expression is true if the result has an odd integer value, a character string value that begins with the letters Y, y, T, or t, or an odd numeric string value.

The expression is false if the result has an even integer value, a character string value that begins with any letter except Y, y, T, or t, or an even numeric string value.

Examples

1.
```
$ COUNT = 0
$ LOOP:
$ COUNT = COUNT + 1
    .
    .
    .
$ IF COUNT .LE. 10 THEN GOTO LOOP
$ EXIT
```

This example shows how to establish a loop in a command procedure, using a symbol named COUNT and an IF statement. The IF statement checks the value of COUNT and performs an EXIT command when the value of COUNT is greater than 10.

2.
```
$ IF P1 .EQS. "" THEN GOTO DEFAULT
$ IF (P1 .EQS. "A") .OR. (P1 .EQS. "B") THEN GOTO 'P1'
$ WRITE SYS$OUTPUT "Unrecognized parameter option ''P1' "
$ EXIT
$ A:       !  Process option a
 .
 .
 .
$ EXIT
$ B:       !  Process option b
 .
 .
 .
$ EXIT
$ DEFAULT: !  Default processing
 .
 .
 .
$ EXIT
```

This example shows a command procedure that tests whether a parameter was passed. The GOTO command passes control to the label specified as the parameter.

If the procedure is executed with a parameter, the procedure uses that parameter to determine the label to branch to. For example:

```
@TESTCOM A
```

When the procedure executes, it determines that P1 is not null, and branches to the label A. Note that the EXIT command causes an exit from the procedure before the label B.

3.
```
$  SET NOON
 .
 .
 .
$  LINK CYGNUS,DRACO,SERVICE/LIBRARY
$ IF $STATUS
$ THEN
$  RUN CYGNUS
$ ELSE
$   WRITE SYS$OUTPUT "LINK FAILED"
$ ENDIF
$ EXIT
```

This command procedure uses the SET NOON command to disable error checking by the command procedure. After the LINK command, the IF command tests the value of the reserved global symbol $STATUS. If the value of $STATUS indicates that the LINK command succeeded, then the program CYGNUS is run. If the LINK command returns an error status value, the command procedure issues a message and exits.

INITIALIZE

Formats a disk or magnetic tape volume, writes a label on the volume, and leaves the disk empty except for the system files containing the structure information. All former contents of the disk are lost.

Requires VOLPRO (volume protection) privilege for most INITIALIZE command operations.

Format

INITIALIZE device-name[:] volume-label

Parameters

device-name[:]

Specifies the name of the device on which the volume to be initialized is physically mounted.

The device does not have to be allocated currently; however, allocating the device before initializing it is the recommended practice.

volume-label

Specifies the identification to be encoded on the volume. For a disk volume, you can specify a maximum of 12 ANSI characters; for a magnetic tape volume, you can specify a maximum of 6 alphanumeric characters. Letters are automatically changed to uppercase. DIGITAL strongly recommends that a disk volume label should only consist of alphanumeric characters, dollar signs ($), underscores (_), and hyphens (-).

To use ANSI ‘‘a’’ characters on the volume label on magnetic tape, you must enclose the volume name in quotation marks (‘‘ ’’). For an explanation of ANSI ‘‘a’’ characters, see the description of the /LABEL qualifier.

Description

The default format for disk volumes in the OpenVMS operating system is called the Files–11 On-Disk Structure Level 2. The default for magnetic tape volumes is based on Level 3 of the ANSI standard for magnetic tape labels and file structure for informational interchange (ANSI X3.27-1978).

The INITIALIZE command can also initialize disk volumes in the Files–11 On-Disk Structure Level 1 format.

You must have VOLPRO privilege to initialize a volume, except in the following cases:

- A blank disk or magnetic tape volume; that is, a volume that has never been written
- A disk volume that is owned by your current user identification code (UIC) or by the UIC [0,0]
- A magnetic tape volume that allows write (W) access to your current UIC that was not protected when it was initialized

After the volume is initialized and mounted, the SET SECURITY command may be used to modify the security profile.

When the INITIALIZE command initializes a magnetic tape volume, it always attempts to read the volume. A blank magnetic tape can sometimes cause unrecoverable errors, such as the following:

- An invalid volume number error message:

  ```
  %INIT-F-VOLINV, volume is invalid
  ```

- A runaway magnetic tape (this frequently occurs with new magnetic tapes that have never been written or that have been run through verifying machines). You can stop a runaway magnetic tape only by setting the magnetic tape drive off line and by then putting it back on line.

If this type of unrecoverable error occurs, you can initialize successfully a magnetic tape by repeating the INITIALIZE command from an account that has VOLPRO (volume protection) privilege and by specifying the following qualifier in the command:

/OVERRIDE=(ACCESSIBILITY,EXPIRATION)

This qualifier ensures that the INITIALIZE command does not attempt to verify any labels on the magnetic tape.

If you have VOLPRO privilege, the INITIALIZE command initializes a disk without reading the ownership information. If you do not have VOLPRO privilege, the INITIALIZE command checks the ownership of the volume before initializing the disk. A blank disk or a disk with an incorrect format can sometimes cause a fatal drive error. If a blank disk or a disk with an incorrect format causes this type of error, you can initialize a disk successfully by repeating the INITIALIZE command with the /DENSITY qualifier from an account that has VOLPRO privilege.

Many of the INITIALIZE command qualifiers allow you to specify parameters that can maximize input/output (I/O) efficiency.

Qualifiers

/ACCESSED=number-of-directories

Affects Files–11 On-Disk Structure Level 1 disks *only*.

Specifies that, for disk volumes, the number of directories allowed in system space must be a value from 0 to 255. The default value is 3.

/BADBLOCKS=(area[,...])

Specifies, for disk volumes, faulty areas on the volume. The INITIALIZE command marks the areas as allocated so that no data is written in them.

Possible formats for area are as follows:

lbn[:count]	Logical block number (LBN) of the first block and optionally a block count beginning with the first block, to be marked as allocated
sec.trk.cyl[:cnt]	Sector, track, and cylinder of the first block, and optionally a block count beginning with the first block, to be marked as allocated

All media supplied by Digital and supported on the OpenVMS operating system, except diskettes and TU58 cartridges, are factory formatted and contain bad block data. The Bad Block Locator utility (BAD) or the diagnostic formatter EVRAC can be used to refresh the bad block data or to construct it for the media

exceptions above. The /BADBLOCKS qualifier is necessary only to enter bad blocks that are not identified in the volume's bad block data.

DIGITAL Storage Architecture (DSA) disks (for example, disks attached to UDA-50 and HSC50 controllers) have bad blocks handled by the controller, and appear logically perfect to the file system.

For information on how to run BAD, see the *OpenVMS Bad Block Locator Utility Manual*.

/CLUSTER_SIZE=number-of-blocks

Defines, for disk volumes, the minimum allocation unit, in blocks. The maximum size you can specify for a volume is one-hundredth the size of the volume; the minimum size you can specify is calculated with the following formula:

$$\frac{disk\ size(number\ of\ blocks)}{255 * 4096}$$

For Files–11 On-Disk Structure Level 2 disks, the VMScluster size default depends on the disk capacity; disks with less than 50,000 have a default of 1. Disks that are larger than 50,000 have a default of 3, unless this default would break the formula. In this case the minimum value allowed by the equation above is applied.

For Files–11 On-Disk Structure Level 1 disks, the VAXcluster size must always be 1.

/DATA_CHECK[=(option[,...])]

Checks all read and write operations on the disk. By default, no data checks are made. Specify one or both of the following options:

READ	Checks all read operations.
WRITE	Checks all write operations; default if only the /DATA_CHECK qualifier is specified.

To override the checking you specify at initialization for disks, enter a MOUNT command to mount the volume.

/DENSITY=density-value

The /DENSITY qualifier is not applicable to any TK or TF tape device.

Valid density values are as follows:

800	6250	HD	SINGLE
1600	DD	ED	DOUBLE

To format a diskette on RXnn diskette drives, use the INITIALIZE/DENSITY command. Specify the density at which the diskette is to be formatted as follows:

Diskette	Density value
RX02	SINGLE or DOUBLE
RX33	DOUBLE
RX26	ED

If you do not specify a density value for a diskette being initialized on a drive, the system leaves the volume at the density to which the volume was last formatted.

Note

Diskettes formatted in double density cannot be read or written by the console block storage device (an RX01 drive) of a VAX–11/780 until they have been reformatted in single density.

RX33 diskettes cannot be read from or written to by RX50 disk drives. RX50 diskettes can be read from and written to by RX33 disk drives; they cannot be formatted by RX33 disk drives.

For magnetic tape volumes, specifies the density in bits per inch (bpi) at which the magnetic tape is to be written.

For magnetic tape volumes, the density value specified can be 800 bpi, 1600 bpi, or 6250 bpi, as long as the density is supported by the magnetic tape drive. If you do not specify a density value for a blank magnetic tape, the system uses a default density of the highest value allowed by the tape drive. If the drive allows 6250-, 1600-, and 800-bpi operation, the default density is 6250 bpi. If you do not specify a density value for a magnetic tape that has been previously written, the system uses the density of the first record on the volume. If the record is unusually short, the density value will not default.

/DIRECTORIES=number-of-entries

Specifies, for disk volumes, the number of entries to preallocate for user directories. The number of entries must be an integer between 16 and 16000. The default value is 16.

/ERASE
/NOERASE (default)

Physically destroys deleted data by writing over it. Controls the data security erase (DSE) operation on the volume before initializing it. The /ERASE qualifier applies to Files–11 On-Disk Structure Level 2 disk and ANSI magnetic tape volumes, and is valid for magnetic tape devices that support the hardware erase function, such as TU78 and MSCP magnetic tapes.

If you specify the /ERASE qualifier, a DSE operation is performed on the volume. For disk devices, the ERASE volume attribute is set. In effect, each file on the volume is erased when it is deleted.

Note that the amount of time taken by the DSE operation depends on the volume size; the INITIALIZE/ERASE command is always slower than the INITIALIZE /NOERASE command.

/EXTENSION=number-of-blocks

Specifies, for disk volumes, the number of blocks to use as a default extension size for all files on the volume. The extension default is used when a file increases to a size greater than its initial default allocation during an update. For Files–11 On-Disk Structure Level 2 disks, the value for the *number-of-blocks* parameter can range from 0 to 65,535. The default value is 5. For Files–11 On-Disk Structure Level 1 disks, the value can range from 0 to 255.

The OpenVMS operating system uses the default volume extension only if no different extension has been set for the file and no default extension has been set for the process by using the SET RMS_DEFAULT command.

/FILE_PROTECTION=code

Affects Files–11 On-Disk Structure Level 1 disks *only*.

Defines for disk volumes the default protection to be applied to all files on the volume.

Specify the code according to the standard syntax rules described in the *OpenVMS Guide to System Security*. Any attributes not specified are taken from the current default protection.

Note that this attribute is not used when the volume is being used on an OpenVMS system, but is provided to control the process's use of the volume on RSX–11M systems. OpenVMS systems always use the default file protection. Use the SET PROTECTION/DEFAULT command to change the default file protection.

/GROUP

Used in conjunction with the /NOSHARE qualifier to create a group volume. The group volume allows access by system (S), owner (O), and group (G) accessors. The protection is (S:RWCD,O:RWCD,G:RWCD,W).

The owner user identification code (UIC) of the volume defaults to your group number and a member number of 0.

/HEADERS=number-of-headers

Specifies, for disk volumes, the number of file headers to be allocated for the index file. The minimum and default value is 16. The maximum is the value set with the /MAXIMUM_FILES qualifier.

This qualifier is useful when you want to create a number of files and want to streamline the process of allocating space for that number of file headers. If you do not specify this qualifier, the file system dynamically allocates space as it is needed for new headers on the volume.

Note

The default value for the /HEADERS qualifier is generally insufficient for ODS-2 disks. To improve performance and avoid SYSTEM-F-HEADERFULL errors, Digital recommends that you set this value to be approximately the number of files that you anticipate having on your disk. However, grossly overestimating this value will result in wasted disk space.

The /HEADERS qualifier controls how much space is initially allocated to INDEXF.SYS for headers. Each file on a disk requires at least one file header and each header occupies one block within INDEXF.SYS. Files that have many Access Control Entries (ACE) or are very fragmented may use more than one header.

The default value of 16 leaves room for less than 10 files to be created before INDEXF.SYS must extend. Therefore, try to estimate the total number of files that will be created on the disk and specify it here. This will improve disk access performance. Over-estimating the value may lead to wasted disk space. This value cannot be changed without reinitializing the volume.

INDEXF.SYS is limited as to how many times it may extend. When the map area in its header (where the retrieval pointers are stored) becomes full, file creation fails with the message "SYSTEM-W-HEADERFULL."

/HIGHWATER (default)
/NOHIGHWATER
Affects Files–11 On-Disk Structure Level 2 disks *only*.

Sets the file high-water mark (FHM) volume attribute, which guarantees that users cannot read data that they have not written. You cannot specify the /NOHIGHWATER qualifier for magnetic tape.

The /NOHIGHWATER qualifier disables FHM for a disk volume.

/HOMEBLOCKS=option
This qualifier applies only to Files–11 ODS–2 volumes.

It specifies where the volume's homeblock and spare copy of the homeblock are placed on disk. The value of **option** can be one of the following:

- GEOMETRY

  Causes the homeblocks to be placed at separate locations on disk, to protect against failure of a disk block. Placement depends on the reported geometry of the disk.

- FIXED

  Causes the homeblocks to be placed at separate fixed locations on the disk; this is the default. Placement is independent of the reported geometry of the disk. This caters for disks that report different geometries according to which type of controller they are attached to.

- CONTIGUOUS

  Causes the homeblocks to be placed contiguously at the start of the disk. This allows container file systems to maximize the amount of contiguous space on the disk, when used with the /INDEX=BEGINNING qualifier.

/INDEX=position
Specifies the location of the index file for the volume's directory structure. Possible positions are as follows:

BEGINNING	Beginning of the volume
MIDDLE	Middle of the volume (default)
END	End of the volume
BLOCK:*n*	Beginning of the logical block specified by *n*

/LABEL=option
Defines characteristics for the magnetic tape volume label, as directed by the included option. The available options are as follows:

- OWNER_IDENTIFIER:''(14 ANSI characters)''

  Allows you to specify the Owner Identifier field in the volume label. The field specified can accept up to 14 ANSI characters.

- VOLUME_ACCESSIBILITY:''character''

  Specifies the character to be written in the volume accessibility field of the OpenVMS ANSI volume label VOL1 on an ANSI magnetic tape. The character may be any valid ANSI ''a'' character. This set of characters includes numeric characters, uppercase letters, and any one of the following nonalphanumeric characters:

  ```
  ! " % ' ( ) * + , - . / : ; < = > ?
  ```

By default, the OpenVMS operating system provides a routine that checks this field in the following manner.

- If the magnetic tape was created on a version of the OpenVMS operating system that conforms to Version 3 of ANSI, then this option must be used to override any character other than an ASCII space.
- If a protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, then this option must be used to override any character other than an ASCII 1.

If you specify any character other than the default, you must specify the /OVERRIDE=ACCESSIBILITY qualifier on the INITIALIZE and MOUNT commands in order to access the magnetic tape.

/MAXIMUM_FILES=n

Restricts the maximum number of files that the volume can contain. The /MAXIMUM_FILES qualifier overrides the default value, which is calculated as follows:

$$\frac{volume\ size\ in\ blocks}{(cluster\ factor + 1) * 2}$$

The maximum size you can specify for any volume is as follows:

$$\frac{volume\ size\ in\ blocks}{(cluster\ factor + 1)}$$

The minimum value is 0. Note that the maximum can be increased only by reinitializing the volume.

Note

The MAXIMUM_FILES qualifier does not reserve or create space for new file headers on a volume. The file system dynamically allocates space as it is needed for new headers.

/MEDIA_FORMAT=[NO]COMPACTION

Controls whether data records are automatically compacted and blocked together on any device that supports data compaction. Data compaction and record blocking increase the amount of data that can be stored on a single tape cartridge.

Note that once data compaction or non-compaction has been selected for a given cartridge, that same status applies to the entire cartridge.

/OVERRIDE=(option[,...])

Requests the INITIALIZE command to ignore data on a magnetic tape volume that protects it from being overwritten. You can specify one or more of the following options:

ACCESSIBILITY	(For magnetic tapes only.) If the installation allows, this option overrides any character in the Accessibility field of the volume. The necessity of this option is defined by the installation. That is, each installation has the option of specifying a routine that the magnetic tape file system will use to process this field. By default, OpenVMS provides a routine that checks this field in the following manner. If the magnetic tape was created on a version of OpenVMS that conforms to Version 3 of ANSI, this option must be used to override any character other than an ASCII space. If a protection is specified and the magnetic tape conforms to an ANSI standard that is higher than Version 3, this option must be used to override any character other than an ASCII 1. To use the ACCESSIBILITY option, you must have the user privilege VOLPRO or be the owner of the volume.
EXPIRATION	(For magnetic tapes only.) Allows you to write to a tape that has not yet reached its expiration date. You may need to do this for magnetic tapes that were created before VAX/VMS Version 4.0 on Digital operating systems using the D% format in the volume Owner Identifier field. You must have the user privilege VOLPRO to override volume protection, or your UIC must match the UIC written on the volume.
OWNER_IDENTIFIER	Allows you to override the processing of the Owner Identifier field of the volume label.

If you specify only one option, you can omit the parentheses.

To initialize a volume that was initialized previously with the /PROTECTION qualifier, your UIC must match the UIC written on the volume or you must have VOLPRO privilege.

You can initialize a volume previously initialized with /PROTECTION if you have control access.

/OWNER_UIC=uic

Specifies an owner user identification code (UIC) for the volume. The default is your default UIC. Specify the UIC using standard UIC format as described in the *OpenVMS Guide to System Security*.

For magnetic tapes, no UIC is written unless protection on the magnetic tape is specified. If protection is specified, but no owner UIC is specified, your current UIC is assigned ownership of the volume.

/PROTECTION=(ownership[:access][,...])

Applies the specified protection to the volume.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), write (W), create (C), or delete (D).

The default is your default protection. Note that the /GROUP, /SHARE, and /SYSTEM qualifiers can also be used to define protection for disk volumes.

For magnetic tape, the protection code is written to an OpenVMS specific volume label. The system applies only read (R) and write (W) access restrictions; create and delete (D) access are meaningless. Moreover, the system and the owner are always given both read (R) and write (W) access to magnetic tapes, regardless of the protection code you specify.

For more information on specifying protection code, see the *OpenVMS Guide to System Security*. Any attributes not specified are taken from the current default protection.

When you specify a protection code for an entire disk volume, the access type E (execute) indicates create access.

/SHARE (default)
/NOSHARE
Permits all categories of access by all categories of ownership. The /NOSHARE qualifier denies access to group (unless the /GROUP qualifier is also specified) and world processes.

/SIZE=n
Specifies the size of the DECram disk (device type DT$_RAM_DISK) to be allocated from available memory. The size of the device is created at disk initialization time. Note that **n** cannot exceed 524,280 blocks. To deallocate space, specify /SIZE=0. All resources specifically allocated to the DECram disk are returned to the system.

/STRUCTURE=level
Specifies whether the volume should be formatted in Files–11 On-Disk Structure Level 1 or 2 (the default). Structure Level 1 is incompatible with the /DATA_ CHECK and /CLUSTER_SIZE qualifiers. The default protection for a Structure Level 1 disk is full access to system, owner, and group, and read (R) access to all other users.

Note that Alpha does not support File-11 On-Disk Structure Level 1 disks. Specifying 1 on Alpha results in an error.

/SYSTEM
Requires a system UIC or SYSPRV (system privilege) privilege.

Defines a system volume. The owner UIC defaults to [1,1]. Protection defaults to complete access by all ownership categories, except that only system processes can create top-level directories.

/USER_NAME=name
Specifies a user name to be associated with the volume. The name must be 1 to 12 alphanumeric characters. The default is your user name.

/VERIFIED
/NOVERIFIED
Indicates whether the disk has bad block data on it. Use the /NOVERIFIED qualifier to ignore bad block data on the disk. The default is the /VERIFIED qualifier for disks with 4096 blocks or more and the /NOVERIFIED qualifier for disks with less than 4096 blocks.

/WINDOWS=n
Specifies the number of mapping pointers (used to access data in the file) to be allocated for file windows. The value can be an integer in the range of 7 to 80. The default is 7.

Examples

1.
```
$ INITIALIZE/USER_NAME=CPA $FLOPPY1 ACCOUNTS
```

   Initializes the volume on $FLOPPY1, labels the volume ACCOUNTS, and gives the volume a user name of CPA.

2.
```
$ ALLOCATE DMA2:  TEMP
  _DMA2: ALLOCATED
$ INITIALIZE  TEMP:  BACK_UP_FILE
$ MOUNT  TEMP:  BACK_UP_FILE
%MOUNT-I-MOUNTED, BACK_UP_FILE mounted on _DMA2:
$ CREATE/DIRECTORY  TEMP:[ARCHIE]
```

   The previous sequence of commands shows how to initialize an RK06/RK07 volume. First, the device is allocated, to ensure that no one else can access it. Then, when the volume is physically mounted on the device, the INITIALIZE command initializes it. When the volume is initialized, the MOUNT command makes the file structure available. Before you can place any files on the volume, you must create a directory, as shown by the CREATE/DIRECTORY command.

3.
```
$ ALLOCATE MT:
  _MTB1:  ALLOCATED
$ INITIALIZE MTB1:  SOURCE
$ MOUNT MTB1:  SOURCE
%MOUNT-I-MOUNTED, SOURCE mounted on _MTB1:
$ COPY *.FOR  MTB1:
$ DIRECTORY MTB1:
   .
   .
   .
$ DISMOUNT MTB1:
```

   These commands show the procedure necessary to initialize a magnetic tape. After allocating a drive, the magnetic tape is loaded on the device, and the INITIALIZE command writes the label SOURCE on it. Then, the MOUNT command mounts the magnetic tape so that files can be written on it.

4.
```
$ BACKUP filespec MUA0: ... /MEDIA_FORMAT=NOCOMPACTION-
_$/REWIND
```

   This example creates a BACKUP tape with compaction and record blocking disabled.

INITIALIZE/QUEUE

Creates or initializes queues. You use this command to create queues and to assign them names and options. The /BATCH qualifier is required to create a batch queue.

Requires OPER (operator) privilege to create queues and manage (M) access to modify queues.

Format

INITIALIZE/QUEUE queue-name[:]

Parameter

queue-name[:]
Specifies the name of an execution queue or a generic queue. The queue name may be a string of 1 to 31 characters. The character string can include any uppercase and lowercase letters, digits, the dollar sign ($), and the underscore (_), and must include at least one alphabetic character.

Description

Use the INITIALIZE/QUEUE command to create a queue or to change the options of an existing queue that is stopped.

Normally you create output and batch queues by entering the necessary INITIALIZE/QUEUE commands when you set up your system or VMScluster. Later, you can use the INITIALIZE/QUEUE command to create additional queues as they are needed. When you create a queue with the INITIALIZE/QUEUE command, information about the queue is stored in the queue database.

To create and start the queue at the same time, you can use the INITIALIZE /QUEUE/START command. If you want to create the queue only and start it at another time, you can enter only the INITIALIZE/QUEUE command. Later you can enter the START/QUEUE command to begin queue operations.

You can use the INITIALIZE/QUEUE, START/QUEUE, and SET QUEUE commands to change queue options; as you change queue options, information about the queue in the queue database is updated.

You can use the INITIALIZE and START commands only on stopped queues. To change options on a running queue, use the SET QUEUE command. To change queue options that cannot be altered with the SET QUEUE command, use the following procedure:

1. Stop the queue with the STOP/QUEUE/NEXT command.
2. Restart the queue with the START/QUEUE or the INITIALIZE/QUEUE /START command, specifying the appropriate qualifiers for the options you desire.

   Any qualifiers that you do not specify remain as they were when the queue was previously initialized, started, or set.

Note that initializing an existing queue does not delete any current jobs in that queue. Any new queue settings established by the new INITIALIZE/QUEUE command affect all jobs waiting in the queue or subsequently entering the queue. Any jobs that are executing in the queue when it is stopped complete their execution under the old settings.

The following qualifiers apply to generic and execution queues:

/OWNER_UIC
/PROTECTION
/[NO]RETAIN
/[NO]START
/NAME_OF_MANAGER

The following qualifiers apply to all types of execution queues:

/AUTOSTART_ON
/BASE_PRIORITY
/[NO]CHARACTERISTICS
/[NO]ENABLE_GENERIC
/[NO]NO_INITIAL_FF
/ON
/WSDEFAULT
/WSEXTENT
/WSQUOTA

The following qualifiers apply only to batch execution queues:

/CPUDEFAULT
/CPUMAXIMUM
/[NO]DISABLE_SWAPPING
/JOB_LIMIT

The following qualifiers apply only to printer, terminal, or server execution queues:

/[NO]BLOCK_LIMIT
/[NO]DEFAULT
/FORM_MOUNTED
/[NO]LIBRARY
/[NO]PROCESSOR
/[NO]RECORD_BLOCKING
/[NO]SEPARATE

Types of Queues

There are several different types of queues. In general, queues can be divided into two major classes: generic and execution. When a job is sent to an execution queue, it is executed in that queue. No processing takes place in generic queues. Generic queues hold jobs that will execute on an execution queue.

Generic Queues The following are several types of generic queues:

- Generic batch queue—Holds batch jobs for execution on batch execution queues.
- Generic output queue—Holds jobs for execution on output queues. There are three types of generic output queues:
  - Generic printer queue—Holds print jobs for printing on output execution queues.
  - Generic server queue—Holds jobs for processing on output execution queues.
  - Generic terminal queue—Holds print jobs for printing on output execution queues.

The /GENERIC qualifier designates a queue as a generic queue. You specify the execution queues to which a generic queue feeds jobs in one of two ways:

- You can explicitly name execution queues assigned to the generic queue by including a list of queues with the /GENERIC qualifier.
- You can specify the execution queues that may receive jobs from any generic queue that does not specify an explicit target list by specifying the /ENABLE=GENERIC qualifier when you create the execution queue.

Generic queues, unlike execution queues, are not automatically stopped when the system is shut down or the queue manager is stopped. Therefore, generic queues do not normally need to be restarted each time the system reboots.

Logical Queues

Another type of queue is the logical queue. A logical queue is a special type of generic queue that can place work only into the execution queue specified in the ASSIGN/QUEUE command. The logical queue's relation to an execution queue remains in effect until you enter a DEASSIGN/QUEUE command to negate the assignment.

Execution Queues The following are several types of execution queues:

- Batch execution queue—Executes batch jobs.
- Output execution queue—Processes print output jobs. There are three types of output execution queues:
  - Printer execution queue—Invokes a symbiont to process print jobs for a printer.
  - Server execution queue—Invokes a customer-written symbiont to process jobs.
  - Terminal execution queue—Invokes a symbiont to process print jobs for a terminal printer.

Batch execution queues execute batch jobs. Batch jobs request the execution of one or more command procedures in a batch process.

Output execution queues process print jobs. A print job requests the processing of one or more files by a symbiont executing in a symbiont process. The default system symbiont is designed to print files on hardcopy devices (printers or terminals). Customer-written symbionts can be designed for this or any other file processing activity. Server queues process jobs using the server symbiont specified with the /PROCESSOR qualifier. Server queue symbionts are written by the customer.

Either the /AUTOSTART_ON qualifier or the /ON qualifier designates a queue as an execution queue, and specifies where the queue is to run.

By using the /ON qualifier, you can specify one node (for batch queues) or node and device (for output queues) on which the queue can be started. A queue initialized with the /ON qualifier needs to be started by a command explicitly naming the queue.

You can specify one or more nodes (or nodes and devices) on which the queue can be started by using the /AUTOSTART_ON qualifier. A queue initialized with the /AUTOSTART_ON qualifier is automatically started by the queue manager when any of the queue's nodes have been enabled for autostart by that queue manager.

Autostart Queues An execution queue (either batch or output) can be designated as an autostart queue. Because all of a queue manager's autostart queues on a node can be started with a single command, autostart queues eliminate the need for lengthy queue startup procedures.

In a VMScluster, autostart queues can be set up to run on one of several nodes. If a queue is set up this way, and the node on which the queue is running leaves the cluster, the queue can fail over to another node and remain available to the cluster.

The /AUTOSTART_ON qualifier designates an execution queue as an autostart queue.

Qualifiers

/AUTOSTART_ON=(node::[device][,...])

Designates the queue as an autostart execution queue and specifies the node, or node and device, on which the queue can be located. For batch queues, only *node* is applicable.

In a cluster, you can specify more than one node (or node and device) on which a queue can run, in the preferred order in which nodes should claim the queue. This allows the queue to fail over to another node if the node on which the queue is running leaves the cluster.

When you enter the INITIALIZE/QUEUE command with the /AUTOSTART_ON qualifier, you must initially activate the queue for autostart, either by specifying the /START qualifier with the INITIALIZE/QUEUE command or by entering a START/QUEUE command. However, the queue will not begin processing jobs until the ENABLE AUTOSTART/QUEUES command is entered for a node on which the queue can run.

This qualifier cannot be used in conjunction with the /ON or /GENERIC qualifier. However, if you are reinitializing an existing queue, you can specify the /AUTOSTART_ON qualifier for a queue previously created or started with the /ON qualifier. Doing so overrides the /ON qualifier and makes the queue an autostart queue.

For more information about autostart queues, see the chapter about queues in *OpenVMS System Manager's Manual: Essentials*.

/BASE_PRIORITY=n

Specifies the base process priority at which jobs are initiated from a batch execution queue. By default, if you omit the qualifier, jobs are initiated at the

same priority as the base priority established by DEFPRI at system generation (usually 4). The base priority specifier can be any decimal value from 0 to 15.

You also can specify this qualifier for an output execution queue. In this context the /BASE_PRIORITY qualifier establishes the base priority of the symbiont process when the symbiont process is created.

/BATCH
/NOBATCH (default)
Specifies that you are initializing a batch queue. If you are reinitializing an existing queue, you can use the /BATCH qualifier only if the queue was created as a batch queue.

A batch queue is classified as either an execution queue or a generic queue. By default, the /BATCH qualifier initializes an execution queue. To specify a generic batch queue, use the /GENERIC qualifier together with the /BATCH qualifier.

The /BATCH and /DEVICE qualifiers are mutually exclusive; the /NOBATCH and /NODEVICE qualifiers cannot be used together.

/BLOCK_LIMIT=([lowlim,]uplim)
/NOBLOCK_LIMIT (default)
Limits the size of print jobs that can be processed on an output execution queue. The /BLOCK_LIMIT qualifier allows you to reserve certain printers for certain size jobs. You must specify at least one of the parameters.

The *lowlim* parameter is a decimal number referring to the minimum number of blocks accepted by the queue for a print job. If a print job is submitted that contains fewer blocks than the *lowlim* value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is decreased sufficiently, the job is processed.

The *uplim* parameter is a decimal number referring to the maximum number of blocks that the queue accepts for a print job. If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. After the block limit for the queue is increased sufficiently, the job is processed.

If you specify only an upper limit for jobs, you can omit the parentheses. For example, /BLOCK_LIMIT=1000 means that only jobs with 1000 blocks or less are processed in the queue. To specify only a lower job limit, you must use a null string ("") to indicate the upper specifier. For example, /BLOCK_LIMIT=(500,"") means any job with 500 or more blocks is processed in the queue. You can specify both a lower and upper limit. For example, /BLOCK_LIMIT=(200,2000) means that jobs with less than 200 blocks or more than 2000 blocks are not processed in the queue.

The /NOBLOCK_LIMIT qualifier cancels the previous setting established by the /BLOCK_LIMIT qualifier for that queue.

/CHARACTERISTICS=(characteristic[,...])
/NOCHARACTERISTICS (default)
Specifies one or more characteristics for processing jobs on an execution queue. If you specify only one characteristic, you can omit the parentheses. If a queue does not have all the characteristics that have been specified for a job, the job remains pending. Each time you specify the /CHARACTERISTICS qualifier, all previously set characteristics are cancelled. Only the characteristics specified with the qualifier are established for the queue.

Queue characteristics are installation specific. The *characteristic* parameter can be either a value from 0 to 127 or a characteristic name that has been defined by the DEFINE/CHARACTERISTIC command.

The /NOCHARACTERISTICS qualifier cancels any settings previously established by the /CHARACTERISTICS qualifier for that queue.

/CLOSE

Prevents jobs from being entered in the queue through PRINT or SUBMIT commands or as a result of requeue operations. To allow jobs to be entered, use the /OPEN qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled). When a queue is marked closed, jobs executing continue to execute. Jobs pending in the queue continue to be candidates for execution.

/CPUDEFAULT=time

Defines the default CPU time limit for all jobs in this batch execution queue. You can specify time as delta time, 0, INFINITE, or NONE (default). You can specify up to 497 days of delta time.

If the queue does not have a specified CPUMAXIMUM time limit and the value established in the user authorization file (UAF) has a specified CPU time limit of NONE, either the value 0 or the keyword INFINITE allows unlimited CPU time. If you specify NONE, the CPU time value defaults to the value specified either in the UAF or by the SUBMIT command (if included). CPU time values must be greater than or equal to the number specified by the system parameter PQL_MCPULM. The time cannot exceed the CPU time limit set by the /CPUMAXIMUM qualifier. For information on specifying delta time, see the *OpenVMS User's Manual*. For more information on specifying CPU time limits, see Table DCLI–1.

/CPUMAXIMUM=time

Defines the maximum CPU time limit for all jobs in a batch execution queue. You can specify time as delta time, 0, INFINITE, or NONE (default). You can specify up to 497 days of delta time.

The /CPUMAXIMUM qualifier overrides the time limit specified in the user authorization file (UAF) for any user submitting a job to the queue. Either the value 0 or the keyword INFINITE allows unlimited CPU time. If you specify NONE, the CPU time value defaults to the value specified either in the UAF or by the SUBMIT command (if included). CPU time values must be greater than or equal to the number specified by the system parameter PQL_MCPULM.

For information on specifying delta times, see the *OpenVMS User's Manual*. For more information on specifying CPU time limits, see Table DCLI–1.

A CPU time limit for processes is specified by each user record in the system UAF. You also can specify the following: a default CPU time limit or a maximum CPU time limit for all jobs in a given queue, or a default CPU time limit for individual jobs in the queue. Table DCLI–1 shows the action taken for each value specified and possible combinations of specifications.

Table DCLI–1 CPU Time Limit Specifications and Actions

CPU Time Limit Specified by the SUBMIT Command?	Default CPU Time Limit Specified for the Queue?	Maximum CPU Time Limit Specified for the Queue?	Action Taken
No	No	No	Use the UAF value.
Yes	No	No	Use the smaller of SUBMIT command and UAF values.
Yes	Yes	No	Use the smaller of SUBMIT command and UAF values.
Yes	No	Yes	Use the smaller of SUBMIT command and queue's maximum values.
Yes	Yes	Yes	Use the smaller of SUBMIT command and queue's maximum values.
No	Yes	Yes	Use the smaller of queue's default and maximum values.
No	No	Yes	Use the maximum value.
No	Yes	No	Use the smaller of UAF and queue's default values.

/DEFAULT=(option[,...])
/NODEFAULT

Establishes defaults for certain options of the PRINT command. Defaults are specified by the list of options. If you specify only one option, you can omit the parentheses. After you set an option for the queue with the /DEFAULT qualifier, you do not have to specify that option in your PRINT command. If you do specify these options in your PRINT command, the values specified with the PRINT command override the values established for the queue with the /DEFAULT qualifier.

You cannot use the /DEFAULT qualifier with the /GENERIC qualifier.

Possible options are as follows:

[NO]BURST[=keyword]	Controls whether two file flag pages with a burst bar between them are printed preceding output. If you specify the value ALL (default), these flag pages are printed before each file in the job. If you specify the value ONE, these flag pages are printed once before the first file in the job.
[NO]FEED	Controls whether a form feed is inserted automatically at the end of a page.
[NO]FLAG[=keyword]	Controls whether a file flag page is printed preceding output. If you specify the value ALL (default), a file flag page is printed before each file in the job. If you specify the value ONE, a file flag page is printed once before the first file in the job.

FORM=type	Specifies the default form for an output execution queue. If a job is submitted without an explicit form definition, this form is used to process the job. If no form type is explicitly specified with the FORM keyword, the system assigns the form DEFAULT to the queue. See also the description of the /FORM_MOUNTED=type qualifier.
[NO]TRAILER[=keyword]	Controls whether a file trailer page is printed following output. If you specify the value ALL (default), a file trailer page is printed after each file in the job. If you specify the value ONE, a trailer page is printed once after the last file in the job.

When you specify the BURST option for a file, the [NO]FLAG option does not add or subtract a flag page from the two flag pages that are printed preceding the file.

For information on establishing mandatory queue options, see the description of the /SEPARATE qualifier. For more information on specifying default queue options, see the chapter on queues in the *OpenVMS System Manager's Manual*.

/DESCRIPTION=string
/NODESCRIPTION (default)

Specifies a string of up to 255 characters used to provide operator-supplied information about the queue.

Enclose strings containing lowercase letters, blanks, or other nonalphanumeric characters (including spaces) in quotation marks (‘‘ ’’).

The /NODESCRIPTION qualifier removes any descriptive text that may be associated with the queue.

/DEVICE[=option]
/NODEVICE

Specifies that you are initializing an output queue of a particular type. If you are reinitializing an existing queue, you can use the /DEVICE qualifier only if the queue was created as an output queue. Possible options are as follows:

PRINTER	Indicates a printer queue.
SERVER	Indicates a server queue. A server queue is controlled by the user-modified or user-written symbiont specified with the /PROCESSOR qualifier.
TERMINAL	Indicates a terminal queue.

If you specify the /DEVICE qualifier without a queue type, the /DEVICE=PRINTER qualifier is used by default.

An output queue is classified as either an execution or generic queue. By default, the /DEVICE qualifier initializes an execution queue of the designated type. To specify a generic printer, server, or terminal queue, use the /GENERIC qualifier with the /DEVICE qualifier.

You specify the queue type with the /DEVICE qualifier for informational purposes. When an output execution queue is started, the symbiont associated with the queue determines the actual queue type. The standard symbiont examines device characteristics to establish whether the queue should be marked as printer or terminal. By convention, user-modified and user-written symbionts mark the

queue as a server queue. The device type of a generic queue need not match the device type of its execution queues.

The /DEVICE and /BATCH qualifiers are mutually exclusive; the /NODEVICE and /NOBATCH qualifiers cannot be used together.

/DISABLE_SWAPPING
/NODISABLE_SWAPPING (default)
Controls whether batch jobs executed from a queue can be swapped in and out of memory.

/ENABLE_GENERIC (default)
/NOENABLE_GENERIC
Specifies whether files queued to a generic queue that does not specify explicit queue names with the /GENERIC qualifier can be placed in this execution queue for processing. For more information, see the description of the /GENERIC qualifier.

/FORM_MOUNTED=type
Specifies the mounted form for an output execution queue.

If no form type is explicitly specified, the system assigns the form DEFAULT to the queue.

If the stock of the mounted form does not match the stock of the default form, as indicated by the /DEFAULT=FORM qualifier, all jobs submitted to this queue without an explicit form definition enter a pending state and remains pending until the stock of the mounted form of the queue is identical to the stock of the form associated with the job.

If a job is submitted with an explicit form and the stock of the explicit form is not identical to the stock of the mounted form, the job enters a pending state and remains pending until the stock of the mounted form of the queue is identical to the stock of the form associated with the job.

To specify the form type, use either a numeric value or a form name that has been defined by the DEFINE/FORM command. Form types are installation-specific. You cannot use the /FORM_MOUNTED qualifier with the /GENERIC qualifier.

/GENERIC[=(queue-name[,...])]
/NOGENERIC (default)
Specifies a generic queue. Also specifies that jobs placed in this queue can be moved for processing to compatible execution queues. The /GENERIC qualifier optionally accepts a list of target execution queues that have been previously defined. For a generic batch queue, these target queues must be batch execution queues. For a generic output queue, these target queues must be output execution queues, but can be of any type (printer, server, or terminal). For example, a generic printer queue can feed a mixture of printer and terminal execution queues.

If you do not specify any target execution queues with the /GENERIC qualifier, jobs can be moved to any execution queue that (1) is initialized with the /ENABLE_GENERIC qualifier, and (2) is the same type (batch or output) as the generic queue.

To define the queue as a generic batch or output queue, you use the /GENERIC qualifier with either the /BATCH or the /DEVICE qualifier. If you specify neither /BATCH nor /DEVICE on creation of a generic queue, the queue becomes a generic printer queue by default.

You cannot use the /SEPARATE qualifier with the /GENERIC qualifier.

/JOB_LIMIT=n
Indicates the number of batch jobs that can be executed concurrently from the queue. Specify a number in the range 1 to 255. The job limit default value for *n* is 1.

/LIBRARY=filename
/NOLIBRARY
Specifies the file name for the device control library. When you initialize an output execution queue, you can use the /LIBRARY qualifier to specify an alternate device control library. The default library is SYS$LIBRARY:SYSDEVCTL.TLB. You can use only a file name as the parameter of the /LIBRARY qualifier. The system always assumes that the file is located in SYS$LIBRARY and that the file type is TLB.

/NAME_OF_MANAGER=name
Identifies the name of the queue manager to control the queue. Once the queue is created, the queue manager assignment may not be altered.

If the /NAME_OF_MANAGER qualifier is omitted, then the default name SYS$QUEUE_MANAGER is used.

If the INITIALIZE/QUEUE command is used to modify a queue, and that queue is not controlled by the default queue manager, then the name of the controlling queue manager should be specified with /NAME_OF_MANAGER qualifier. Alternately, the logical name SYS$QUEUE_MANAGER can be defined to be the correct queue manager, making that queue manager the default for the current process.

/NO_INITIAL_FF
/NONO_INITIAL_FF (default)
Allows user to specify whether a form feed should be sent to a printer device when a queue starts. To suppress the initial form feed, use the /NO_INITIAL_FF qualifier.

The /NONO_INITIAL_FF qualifier sends a form feed to the output device to ensure the paper is at the top of a page before printing begins.

/ON=[node::]device[:] (printer, terminal, server queue)
/ON=node:: (batch queue)
Specifies the node or device, or both, on which this execution queue is located. For batch execution queues, you can specify only the node name. For output execution queues, you can include both the node name and the device name. By default, a queue executes on the same node from which you start the queue. The default device parameter is the same as the queue name.

The node name is used in VAXcluster systems; it must match the node name specified by the system parameter SCSNODE for the VAX computer on which the queue executes.

You cannot use the /ON qualifier with the /AUTOSTART_ON or /GENERIC qualifier. However, if you are reinitializing an existing queue, you can specify the /ON qualifier for a queue previously created or started with the /AUTOSTART_ ON qualifier. Doing so overrides the /AUTOSTART_ON option and makes the queue a nonautostart queue.

/OPEN (default)
Allows jobs to be entered in the queue through PRINT or SUBMIT commands or as the result of requeue operations. To prevent jobs from being entered in the queue, use the /CLOSE qualifier. Whether a queue accepts or rejects new job entries is independent of the queue's state (such as paused, stopped, or stalled).

/OWNER_UIC=uic
Enables you to change the user identification code (UIC) of the queue. Specify the UIC by using standard UIC format as described in the *OpenVMS Guide to System Security*. The default UIC is [1,4].

/PROCESSOR=filename
/NOPROCESSOR
Allows you to specify your own print symbiont for an output execution queue. You can use any valid file name as a parameter of the /PROCESSOR qualifier. The system supplies the device and directory name SYS$SYSTEM and the file type EXE. If you use this qualifier for an output queue, it specifies that the symbiont image to be executed is SYS$SYSTEM:filename.EXE.

By default, SYS$SYSTEM:PRTSMB.EXE is the symbiont image associated with an output execution queue.

The /NOPROCESSOR qualifier cancels any previous setting established with the /PROCESSOR qualifier and causes SYS$SYSTEM:PRTSMB.EXE to be used.

/PROTECTION=(ownership[:access],...)
Specifies the protection of the queue.

- Specify the *ownership* parameter as system (S), owner (O), group (G), or world (W).
- Specify the *access* parameter as read (R), submit (S), manage (M), or delete (D).

A null access specification means no access. The default protection is (SYSTEM:M, OWNER:D, GROUP:R, WORLD:S). If you include only one protection code, you can omit the parentheses. For more information on specifying protection codes, see the *OpenVMS Guide to System Security*. For more information on controlling queue operations through UIC-based protection, see the chapter on queues in the *OpenVMS System Manager's Manual*.

/RECORD_BLOCKING (default)
/NORECORD_BLOCKING
Determines whether the symbiont can concatenate (or block together) output records for transmission to the output device. If you specify the /NORECORD_BLOCKING qualifier, the symbiont sends each formatted record in a separate I/O request to the output device. For the standard OpenVMS print symbiont, record blocking can have a significant performance advantage over single-record mode.

/RETAIN[=option]
/NORETAIN (default)
Holds jobs in the queue in a retained state after they have executed. The /NORETAIN qualifier enables you to reset the queue to the default. Possible options are as follows:

ALL (default)	Holds all jobs in the queue after execution.

ERROR	Holds in the queue only jobs that complete unsuccessfully.

A user can request a job retention option for a job by specifying the /RETAIN qualifier with the PRINT, SUBMIT, or SET ENTRY command. However, the job retention option you specify for a queue overrides any job retention option requested by a user for a job in that queue.

/SCHEDULE=SIZE (default)
/SCHEDULE=NOSIZE

Specifies whether pending jobs in an output execution queue are scheduled for printing based on the size of the job. When the default qualifier /SCHEDULE=SIZE is in effect, shorter jobs print before longer ones. When the /SCHEDULE=NOSIZE qualifier is in effect, jobs are printed in the order they were submitted, regardless of size.

If you enter this command while there are pending jobs in any queue, its effect on future jobs is unpredictable.

/SEPARATE=(option[,...])
/NOSEPARATE (default)

Specifies the mandatory queue options, or job separation options, for an output execution queue. Job separation options cannot be overridden by the PRINT command.

You cannot use the /SEPARATE qualifier with the /GENERIC qualifier.

The job separation options are as follows:

[NO]BURST	Specifies whether two job flag pages with a burst bar between them are printed at the beginning of each job.
[NO]FLAG	Specifies whether a job flag page is printed at the beginning of each job.
[NO]TRAILER	Specifies whether a job trailer page is printed at the end of each job.
[NO]RESET=(module[,...])	Specifies one or more device control library modules that contain the job reset sequence for the queue. The specified modules from the queue's device control library (by default SYS$LIBRARY:SYSDEVCTL) are used to reset the device at the end of each job. The RESET sequence occurs after any file trailer and before any job trailer. Thus, all job separation pages are printed when the device is in its RESET state.

When you specify the /SEPARATE=BURST qualifier, the [NO]FLAG separation option does not add or subtract a flag page from the two flag pages that are printed preceding the job.

For information on establishing queue options that can be overridden, see the description of the /DEFAULT qualifier.

For more information on specifying mandatory queue options, see the chapter on queues in the *OpenVMS System Manager's Manual*.

/START
/NOSTART (default)
Starts the queue being initialized by the current INITIALIZE/QUEUE command.

For autostart queues, this qualifier activates the queue for autostart. The queue begins processing jobs when autostart is enabled with the ENABLE AUTOSTART /QUEUES command on any node on which the queue can run.

/WSDEFAULT=n
Defines for a batch job a working set default, the default number of physical pages that the job can use.

The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

Specify the value of *n* as a number of 512-byte pagelets on Alpha systems or 512-byte pages on VAX. Note that OpenVMS rounds up this value to the nearest CPU-specific page so that actual amount of physical memory allowed may be larger than the specified amount on Alpha. For further information, see the *OpenVMS System Manager's Manual*.

If you specify 0 or NONE, the working set default value defaults to the value specified in the UAF or by the SUBMIT command (if it includes a WSDEFAULT value).

You also can specify this qualifier for an output execution queue. Used in this context, the /WSDEFAULT qualifier establishes the working set default of the symbiont process for an output execution queue when the symbiont process is created.

For more information about the way a working set default affects batch jobs, see Table DCLI–2.

/WSEXTENT=n
Defines for the batch job a working set extent, the maximum amount of physical memory that the job can use. The job only uses the maximum amount of physical memory when the system has excess free pages. The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

Specify the value of *n* as a number of 512-byte pagelets on Alpha or and 512-byte pages on VAX. Note that OpenVMS rounds up this value to the nearest CPU-specific page so that actual amount of physical memory allowed may be larger than the specified amount on Alpha.

If you specify 0 or NONE, the working set extent value defaults to the value specified in the UAF or by the SUBMIT command (if it includes a WSEXTENT value).

You also can specify this qualifier for an output execution queue. Used in this context, the /WSEXTENT qualifier establishes the working set extent of the symbiont process for an output execution queue when the symbiont process is created.

For more information about the way a working set extent affects batch jobs, see Table DCLI–2.

/WSQUOTA=n
Defines for a batch job a working set quota, the amount of physical memory that is guaranteed to the job.

The value set by this qualifier overrides the value defined in the user authorization file (UAF) of any user submitting a job to the queue.

Specify the value of *n* as a number of 512-byte pagelets on OpenVMS Alpha or 512-byte pages on OpenVMS VAX. OpenVMS rounds up this value to the nearest CPU-specific page so that actual amount of physical memory allowed may be larger than the specified amount on OpenVMS Alpha. For further information, see the *OpenVMS System Manager's Manual*.

If you specify 0 or NONE, the working set quota value defaults to the value specified in the UAF or by the SUBMIT command (if it includes a WSQUOTA value).

You also can specify this qualifier for an output execution queue. Used in this context, the /WSQUOTA qualifier establishes the working set quota of the symbiont process for an output execution queue when the symbiont process is created.

Working set default, working set quota, and working set extent values are included in each user record in the system UAF. You can specify working set values for individual jobs or for all jobs in a given queue. The decision table (Table DCLI–2) shows the action taken for different combinations of specifications that involve working set values.

Table DCLI–2 Working Set Default, Extent, and Quota Decision

Is the SUBMIT command value specified?	Is the queue value specified?	Action taken
No	No	Use the UAF value.
No	Yes	Use value for the queue.
Yes	Yes	Use smaller of the two values.
Yes	No	Compare specified value with UAF value; use the smaller.

Examples

1.
```
$ INITIALIZE/QUEUE/BATCH/START -
_$ /AUTOSTART_ON=(DATA::, WARF::, DEANNA::) BATCH_1
```

   The INITIALIZE/QUEUE command in this example creates the batch queue BATCH_1, and designates it as an autostart queue capable of executing on node DATA, WARF, or DEANNA. The /START qualifier activates the queue for autostart. The queue will begin executing on the first node (in the list of nodes specified) for which the ENABLE AUTOSTART/QUEUES command is entered.

   If the node on which BATCH_1 is executing is taken out of the VMScluster, the queue will be stopped on that node and will fail over to the first available node in the node list on which autostart is enabled for a queue manager SYS$QUEUE_MANAGER.

As long as autostart is enabled on one of the nodes in the list, this queue will be started and available to execute batch jobs. If all three nodes in the example are shut down or if autostart is disabled, the queue will remain stopped until one of the three nodes in the node list joins the cluster and executes the ENABLE AUTOSTART/QUEUES command.

The ENABLE AUTOSTART/QUEUES and INITIALIZE/QUEUE commands affect only the queues managed by the default queue manager SYS$QUEUE_MANAGER because the /NAME_OF_MANAGER qualifier is not specified.

2.
```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=3 SYS$BATCH
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=1/WSEXTENT=2000 BIG_BATCH
```

In this example, the first INITIALIZE/QUEUE command creates a batch queue called SYS$BATCH that can be used for any batch job. The /JOB_LIMIT qualifier allows three jobs to execute concurrently. The second INITIALIZE/QUEUE command creates a second batch queue called BIG_BATCH that is designed for large jobs. Only one job can execute at a time. The working set extent can be as high as 125 pages on OpenVMS Alpha (on a system with 8KB pages) or 2000 pages on OpenVMS VAX.

3.
```
$ INITIALIZE/QUEUE/START/DEFAULT=(FLAG,TRAILER=ONE)-
_$ /ON=LPA0: LPA0_PRINT
$ INITIALIZE/QUEUE/START/DEFAULT=(FLAG,TRAILER=ONE)-
_$ /BLOCK_LIMIT=(1000,"")/ON=LPB0: LPB0_PRINT
$ INITIALIZE/QUEUE/START/GENERIC=(LPA0_PRINT,LPB0_PRINT) SYS$PRINT
$ INITIALIZE/QUEUE/START/FORM_MOUNTED=LETTER-
_$ /BLOCK_LIMIT=50/ON=TXA5: LQP
```

In this example, the first three INITIALIZE/QUEUE commands set up printer queues. Both queue LPA0_PRINT and LPB0_PRINT are set up to put a flag page before each file within a job and a trailer page after only the last page in a job. In addition, LPB0_PRINT has a minimum block size of 1000. Thus only print jobs larger than 1000 blocks can execute on that queue. SYS$PRINT is established as a generic queue that can direct jobs to either LPA0_PRINT or LPB0_PRINT. Jobs that are too small to run on LPB0_PRINT will be queued from SYS$PRINT to LPA0_PRINT.

The last INITIALIZE/QUEUE command sets up a terminal queue on TXA5. A job queued with a form that has a stock type other than the stock type of form LETTER remains pending in the queue until a form with the same stock type is mounted on the queue, or until the entry is deleted from the queue or moved to another queue. LETTER has been established at this site to indicate special letterhead paper. The block size limit is 50, indicating that this queue is reserved for jobs smaller than 51 blocks.

INQUIRE

Reads a value from SYS$COMMAND (usually the terminal in interactive mode or the next line in the main command procedure) and assigns it to a symbol.

Format

INQUIRE symbol-name [prompt-string]

Parameters

symbol-name

Specifies a symbol consisting of 1 to 255 alphanumeric characters.

prompt-string

Specifies the prompt to be displayed at the terminal when the INQUIRE command is executed. String values are automatically converted to uppercase. Also, any leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space.

Enclose the prompt in quotation marks (‘‘ ’’) if it contains lowercase characters, punctuation, multiple blanks or tabs, or an at sign (@). To denote an actual quotation mark in a prompt-string, enclose the entire string in quotation marks and use quotation marks (‘‘ ’’) within the string.

When the system displays the prompt string at the terminal, it generally places a colon (:) and a space at the end of the string. (See the /PUNCTUATION qualifier.)

If you do not specify a prompt string, the command interpreter uses the symbol name to prompt for a value.

Description

The INQUIRE command displays the prompting message to and reads the response from the input stream established when your process was created. This means that when the INQUIRE command is executed in a command procedure executed interactively, the prompting message is always displayed on the terminal, regardless of the level of nesting of command procedures. Note that input to the INQUIRE command in command procedures will be placed in the RECALL buffer.

When you enter a response to the prompt string, the value is assigned as a character string to the specified symbol. Lowercase characters are automatically converted to uppercase, leading and trailing spaces and tabs are removed, and multiple spaces and tabs between characters are compressed to a single space. To prohibit conversion to uppercase and retain space and tab characters, place quotation marks around the string.

To use symbols or lexical functions when you enter a response to the prompt string, use single quotation marks (‘ ’) to request symbol substitution.

Note that you can also use the READ command to obtain data interactively from the terminal. The READ command accepts data exactly as the user types it; characters are not automatically converted to uppercase and spaces are not compressed. However, symbols and lexical functions will not be translated even if you use apostrophes to request symbol substitution.

INQUIRE

When an INQUIRE command is entered in a batch job, the command reads the response from the next line in the command procedure; if procedures are nested, it reads the response from the first level command procedure. If the next line in the batch job command procedure begins with a dollar sign ($), the line is interpreted as a command, not as a response to the INQUIRE command. The INQUIRE command then assigns a null string to the specified symbol, and the batch job continues processing with the command on the line following the INQUIRE command.

Qualifiers

/GLOBAL
Specifies that the symbol be placed in the global symbol table. If you do not specify the /GLOBAL qualifier, the symbol is placed in the local symbol table.

/LOCAL (default)
Specifies that the symbol be placed in the local symbol table for the current command procedure.

/PUNCTUATION (default)
/NOPUNCTUATION
Inserts a colon and a space after the prompt when it is displayed on the terminal. To suppress the colon and space, specify the /NOPUNCTUATION qualifier.

Examples

1.
```
$ INQUIRE CHECK "Enter Y[ES] to continue"
$ IF .NOT. CHECK THEN EXIT
```

The INQUIRE command displays the following prompting message at the terminal:

```
Enter Y[ES] to continue:
```

The INQUIRE command prompts for a value, which is assigned to the symbol CHECK. The IF command tests the value assigned to the symbol CHECK. If the value assigned to CHECK is true (that is, an odd numeric value, a character string that begins with a T, t, Y, or y, or an odd numeric character string), the procedure continues executing.

If the value assigned to CHECK is false (that is, an even numeric value, a character string that begins with any letter except T, t, Y, or y, or an even numeric character string), the procedure exits.

2.
```
$ INQUIRE COUNT
$ IF COUNT .GT. 10 THEN GOTO SKIP
  .
  .
  .
$ SKIP:
```

The INQUIRE command prompts for a count with the following message:

```
COUNT:
```

Then the command procedure uses the value of the symbol COUNT to determine whether to execute the next sequence of commands or to transfer control to the line labeled SKIP.

3.
```
$ IF P1 .EQS. "" THEN INQUIRE P1 "FILE NAME"
$ FORTRAN 'P1'
```

The IF command checks whether a parameter was passed to the command procedure by checking if the symbol P1 is null; if it is, it means that no parameter was specified, and the INQUIRE command is issued to prompt for the parameter. If P1 was specified, the INQUIRE command is not executed, and the FORTRAN command compiles the name of the file specified as a parameter.

INSTALL

Invokes the Install utility, which enhances the performance of selected executable and shareable images by making them ‘‘known’’ to the system and assigning them appropriate attributes. For a complete description of the Install utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

INSTALL [subcommand] [filespec]

JOB

Identifies the beginning of a batch job submitted through a card reader. Each batch job submitted through the system card reader must be preceded by a JOB card.

JOB cannot be abbreviated.

Format

$ JOB user-name

Parameter

user-name
Identifies the user name under which the job is to be run. Specify the user name as you would during the login procedure.

Description

The JOB card identifies the user submitting the job and is followed by a PASSWORD card giving the password. (Although the PASSWORD card is required, you do not have to use a password on the card if the account has a null password.)

The user name and password are validated by the system authorization file in the same manner as they are validated in the login procedure. The process that executes the batch job is assigned the disk and directory defaults and privileges associated with the user account. If a LOGIN.COM file exists for the specified user name, it is executed at the start of the job.

The end of a batch job is signaled by the EOJ command, by an EOF card (12-11-0-1-6-7-8-9 overpunch), or by another JOB card.

Qualifiers

/AFTER=time
Holds the job until the specified time. If the specified time has already passed, the job is queued for immediate processing.

The time can be specified as either absolute time or a combination of absolute and delta times. For complete information on specifying time values, see the *OpenVMS User's Manual*.

/CHARACTERISTICS=(characteristic[,...])
Specifies one or more characteristics required for processing the job. If you specify only one characteristic, you can omit the parentheses. Codes for characteristics are installation-defined. Use the SHOW QUEUE/CHARACTERISTICS command to see which characteristics are available on your system.

All the characteristics specified for the job must also be specified for the queue that will execute the job. If not, the job remains pending in the queue until the queue characteristics are changed or the entry is deleted with the DELETE /ENTRY command. Users need not specify every characteristic of a queue with the JOB command as long as the ones they specify are a subset of the characteristics set for that queue. The job also runs if no characteristics are specified.

/CLI=filename
Specifies a different command language interpreter (CLI) with which to process the job. The *filename* parameter specifies that the CLI be SYS$SYSTEM:*filename*.EXE. The default CLI is that defined in the user authorization file (UAF).

/CPUTIME=n
Specifies a CPU time limit for the batch job. Time can be specified as delta time, 0, NONE, or INFINITE. (For information on specifying time values, see the *OpenVMS User's Manual*.)

When you need less CPU time than authorized, use the /CPUTIME qualifier to override the base queue value established by the system manager or the value authorized in your UAF. Specify 0 or INFINITE to request an infinite amount of time. Specify NONE when you want the CPU time to default to your UAF value or the limit specified on the queue. Note that you cannot request more time than permitted by the base queue limits or your UAF.

/DELETE (default)
/NODELETE
Controls whether the batch input file is deleted after the job is processed. If you specify the /NODELETE qualifier, the file is saved in the user's default directory under the default name INPBATCH.COM. If you specify the /NAME qualifier, the file name of the batch input file is the same as the job name you supply with the /NAME qualifier.

/HOLD
/NOHOLD (default)
Controls whether or not the job is to be made available for immediate processing.

If you specify the /HOLD qualifier, the job is not released for processing until you specifically release it with the /NOHOLD or the /RELEASE qualifier of the SET QUEUE/ENTRY command.

/KEEP
/NOKEEP (default)
Controls whether the log file is deleted after it is printed. The /NOKEEP qualifier is the default unless you specify the /NOPRINTER qualifier.

/LOG_FILE=filespec
/NOLOG_FILE
Controls whether a log file with the specified name is created for the job or whether a log file is created.

When you use the /LOG_FILE qualifier, the system writes the log file to the file you specify. If you use the /NOLOG_FILE qualifier, no log file is created. If you specify neither form of the qualifier, the log file is written to a file in your default directory that has the same file name as the first command file in the job and a file type of LOG. Using neither the /LOG_FILE nor the /NOLOG_FILE qualifier is the default.

You can use the /LOG_FILE qualifier to specify that the log file be written to a different device. Logical names that occur in the file specification are translated at the time the job is submitted. The process executing the batch job must have access to the device on which the log file will reside.

If you omit the /LOG_FILE qualifier and specify the /NAME qualifier, the log file is written to a file having the same file name as that specified by the /NAME qualifier and the file type LOG.

/NAME=job-name
Specifies a string to be used as the job name and as the file name for both the batch job log file and the command file. The job name must be 1 to 39 alphanumeric characters and must be a valid file name. The default log file name is INPBATCH.LOG; the default command file name is INPBATCH.COM.

/NOTIFY
/NONOTIFY (default)
Controls whether a message is broadcast to any terminal at which you are logged in, notifying you when your job completes or aborts.

/PARAMETERS=(parameter[,...])
Specifies 1 to 8 optional parameters that can be passed to the command procedure. The parameters define values to be equated to the symbols P1 to P8 in the batch job. The symbols are local to the specified command procedure.

If you specify only one parameter, you can omit the parentheses.

The commas (,) delimit individual parameters. If the parameter contains any spaces, special characters or delimiters, or lowercase characters, enclose it in quotation marks (‘‘ ’’). Individual parameters cannot exceed 255 characters.

/PRINTER=queue-name
/NOPRINTER
Controls whether the job log file is queued to the specified queue for printing when the job is complete. The default print queue for the log file is SYS$PRINT.

If you specify the /NOPRINTER qualifier, the /KEEP qualifier is assumed.

/PRIORITY=n
Requires OPER (operator) or ALTPRI (alter priority) privilege to raise the priority above the value of the system parameter MAXQUEPRI.

Specifies the job scheduling priority for the specified job. The value of *n* is an integer from 0 to 255, where 0 is the lowest priority and 255 is the highest.

The default value for the /PRIORITY qualifier is the value of the system parameter DEFQUEPRI. No privilege is needed to set the priority lower than the MAXQUEPRI value.

The /PRIORITY qualifier has no effect on the process priority. The queue establishes the process priority.

/QUEUE=queue-name[:]
Specifies the name of the batch queue in which the job is to be entered. If you do not specify the /QUEUE qualifier, the job is placed in the default system batch job queue, SYS$BATCH.

/RESTART
/NORESTART (default)
Specifies whether the job restarts after a system failure or a STOP/QUEUE/REQUEUE command.

/TRAILING_BLANKS (default)
/NOTRAILING_BLANKS
Controls whether input cards in the card deck are read in card image form or input records are truncated at the last nonblank character. By default, the system does not remove trailing blanks from records read through the card reader. Use the /NOTRAILING_BLANKS qualifier to request that input records be truncated.

/WSDEFAULT=n
Defines a working set default for the batch job; the /WSDEFAULT qualifier overrides the working set size specified in the user authorization file (UAF).

Specify the value of *n* as a number of 512-byte pagelets on Alpha or 512-byte pages on VAX. Note that OpenVMS rounds up this value to the nearest CPU-specific page so that actual amount of physical memory allowed may be larger than the specified amount on Alpha. For further information, see the *OpenVMS System Manager's Manual*. The value *n* can be any integer from 1 to 65,535, 0, or the keyword NONE.

Use this qualifier to impose a value lower than the base queue value established by the system manager or lower than the value authorized in your UAF. A value of 0 or the keyword NONE sets the default value to the value specified either in your UAF or by the working set quota established for the queue. You cannot request a value higher than your default.

/WSEXTENT=n
Defines a working set extent for the batch job; the /WSEXTENT qualifier overrides the working set extent in the UAF.

Specify the value of *n* as a number of 512-byte pagelets on Alpha or 512-byte pages on VAX. Note that OpenVMS rounds up this value to the nearest CPU-specific page so that actual amount of physical memory allowed may be larger than the specified amount on Alpha. For further information, see the *OpenVMS System Manager's Manual*. The value *n* can be any integer from 1 to 65,535, 0, or the keyword NONE.

To impose a lower value, use this qualifier to override the base queue value established by the system manager rather than the value authorized in your UAF. A value of 0 or the keyword NONE sets the default value either to the value specified in the UAF or working set extent established for the queue. You cannot request a value higher than your default.

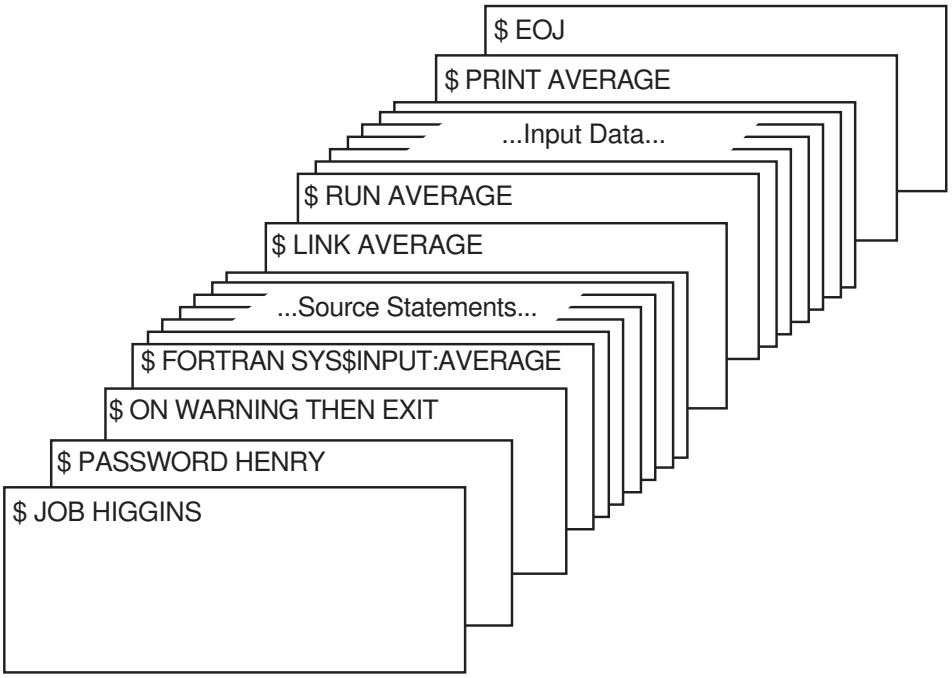
/WSQUOTA=n
Defines the maximum working set size (working set quota) for the batch job; the /WSQUOTA qualifier overrides the value in the UAF.

Specify the value of *n* as a number of 512-byte pagelets on Alpha or 512-byte pages on VAX. Note that OpenVMS rounds up this value to the nearest CPU-specific page so that actual amount of physical memory allowed may be larger than the specified amount on Alpha. For further information, see the *OpenVMS System Manager's Manual*. The value *n* can be any integer from 1 to 65,535, 0, or the keyword NONE.

Use this qualifier to impose a value lower than the base queue value established by the system manager or lower than the value authorized in your UAF. Specify 0 or NONE if you want the working set quota defaulted to either your UAF value or the working set quota specified on the queue. You cannot request a value higher than your default.

Examples

1.

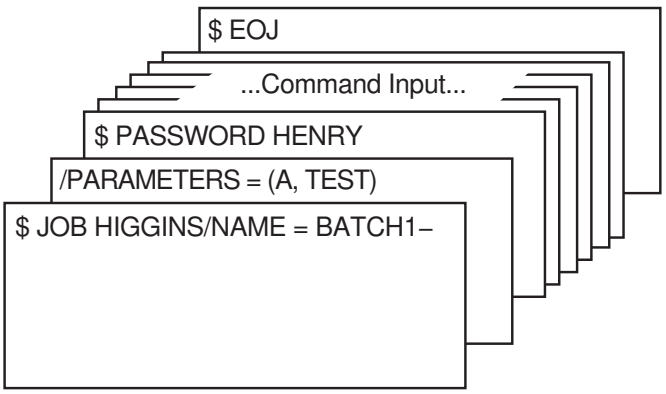


ZK–0787–GE

The JOB and PASSWORD cards identify and authorize the user HIGGINS to enter batch jobs. The command stream consists of a FORTRAN command and FORTRAN source statements to be compiled. The file name AVERAGE following the device name SYS$INPUT provides the compiler with a file name for the object and listing files. The output files are cataloged in user HIGGINS's default directory.

If the compilation is successful, the LINK command creates an executable image and the RUN command executes it. Input for the program follows the RUN command in the command stream. The last command in the job prints the program listing. The last card in the deck contains the EOJ (end of job) command.

2.



ZK–0788–GE

The /NAME qualifier on the JOB card specifies a name for the batch job. When the job completes, the printed log file is identified as BATCH1.LOG. The JOB command is continued onto a second card with the continuation character (-). The /PARAMETERS qualifier defines P1 as A and P2 as TEST. The last card in the deck contains the EOJ (end of job) command.

Lexical Functions

A set of functions that return information about character strings and attributes of the current process.

Description

The command language includes constructs, called lexical functions, that return information about the current process and about arithmetic and string expressions. The functions are called lexical functions because the command interpreter evaluates them during the command input scanning (or lexical processing) phase of command processing.

You can use lexical functions in any context in which you normally use symbols or expressions. In command procedures, you can use lexical functions to translate logical names, to perform character string manipulations, and to determine the current processing mode of the procedure.

The general format of a lexical function is as follows:

```
F$function-name([args,...])
```

where:

F$	Indicates that what follows is a lexical function.
function-name	A keyword specifying the function to be evaluated. Function names can be truncated to any unique abbreviation.
( )	Enclose function arguments, if any. The parentheses are required for all functions, including functions that do not accept any arguments.
args,...	Specify arguments for the function, if any, using integer or character string expressions.

For more information on specifying expressions, see the *OpenVMS User's Manual*.

Table DCLI–3 lists each lexical function and briefly describes the information that each function returns. A detailed description of each function, including examples, is given in the following pages.

Table DCLI–3 Summary of Lexical Functions

Function	Description
F$CONTEXT	Specifies selection criteria for use with the F$PID function.
F$CSID	Returns a VMScluster identification number and updates the context symbol to point to the current position in the system's cluster node list.

(continued on next page)

Table DCLI–3 (Cont.) Summary of Lexical Functions

Function	Description
F$CVSI	Extracts bit fields from character string data and converts the result, as a signed value, to an integer.
F$CVTIME	Retrieves information about an absolute, combination, or delta time string.
F$CVUI	Extracts bit fields from character string data and converts the result, as an unsigned value, to an integer.
F$DEVICE	Returns device names of all devices on a system that meet the specified selection criteria.
F$DIRECTORY	Returns the current default directory name string.
F$EDIT	Edits a character string based on the edits specified.
F$ELEMENT	Extracts an element from a string in which the elements are separated by a specified delimiter.
F$ENVIRONMENT	Obtains information about the DCL command environment.
F$EXTRACT	Extracts a substring from a character string expression.
F$FAO	Invokes the $FAO system service to convert the specified control string to a formatted ASCII output string.
F$FILE_ATTRIBUTES	Returns attribute information for a specified file.
F$GETDVI	Invokes the $GETDVI system service to return a specified item of information for a specified device.
F$GETJPI	Invokes the $GETJPI system service to return accounting, status, and identification information for a process.
F$GETQUI	Invokes the $GETQUI system service to return information about queues, batch and print jobs currently in those queues, form definitions, and characteristic definitions kept in the queue database. †On VAX, also returns information about a queue manager.
F$GETSYI	Invokes the $GETSYI system service to return status and identification information about the local system, or about a node in the local cluster, if your system is part of a cluster.
F$IDENTIFIER	Converts an identifier in named format to its integer equivalent, or vice versa.

†VAX specific

(continued on next page)

Table DCLI–3 (Cont.) Summary of Lexical Functions

Function	Description
F$INTEGER	Returns the integer equivalent of the result of the specified expression.
F$LENGTH	Returns the length of a specified string.
F$LOCATE	Locates a character or character substring within a string and returns its offset within the string.
F$MESSAGE	Returns the message text associated with a specified system status code value.
F$MODE	Shows the mode in which a process is executing.
F$PARSE	Invokes the $PARSE RMS service to parse a file specification and return either the expanded file specification or the particular file specification field that you request.
F$PID	For each invocation, returns the next process identification number in sequence.
F$PRIVILEGE	Returns a value of TRUE or FALSE depending on whether your current process privileges match the privileges listed in the argument.
F$PROCESS	Returns the current process name string.
F$SEARCH	Invokes the $SEARCH RMS service to search a directory file, and returns the full file specification for a file you name.
F$SETPRV	Sets the specified privileges and returns a list of keywords indicating the previous state of these privileges for the current process.
F$STRING	Returns the string equivalent of the result of the specified expression.
F$TIME	Returns the current date and time of day, in the format dd-mmm-yyyy hh:mm:ss.cc.
F$TRNLNM	Translates a logical name and returns the equivalence name string or the requested attributes of the logical name.
F$TYPE	Determines the data type of a symbol.
F$USER	Returns the current user identification code (UIC).
F$VERIFY	Returns the integer 1 if command procedure verification is set on; returns the integer 0 if command procedure verification is set off. The F$VERIFY function also can set new verification states.

F$CONTEXT

Specifies selection criteria for use with the F$PID function. The F$CONTEXT function enables the F$PID function to obtain information about processes from any node in a VMScluster system.

Format

F$CONTEXT(context-type, context-symbol, selection-item, selection-value, value-qualifier)

Return Value

A null string ("").

Arguments

context-type

Specifies the type of context to be built. At present, the only context type available is PROCESS, which is used in constructing selection criteria for F$PID.

context-symbol

Specifies a symbol that DCL uses to refer to the context memory being constructed by the F$CONTEXT function. The function F$PID uses this context symbol to process the appropriate list of process identification (PID) numbers.

Specify the context symbol by using a symbol. The first time you use the F$CONTEXT function in a command procedure, use a symbol that is either undefined or equated to the null string. The symbol created will be a local symbol of type ''PROCESS_CONTEXT''. When the context is no longer valid—that is, when all PIDs have been retrieved by calls to the F$PID function or an error occurs during one of these calls—the symbol no longer has a type of ''PROCESS_CONTEXT''. Then you can use the F$TYPE function in the command procedure to find out if it is necessary to cancel the context.

After setting up the selection criteria, use this context symbol when calling F$PID.

selection-item

Specifies a keyword that tells F$CONTEXT which selection criterion to use. Use only one selection-item keyword per call to F$CONTEXT.

Note

Do not use the NEQ selection value on a list of items because it causes the condition to always be true.

For example:

```
$ EXAMPLE=f$context("PROCESS",CTX,"USERNAME","A*,B*","NEQ")
```

This equation is parsed as ''if the user name is not equal to A* *or* the user name is not equal to B*, then return the process of the users that meet the criteria.'' Because the operand is a logical *or*, the conditions will always be true (any name will be found to be not equal to A* or B*; ALFRED will not be equal to B*; BOB will not be equal to A*).

The following table shows valid selection-item keywords for the PROCESS context type:

Selection Item	Selection Value	Value Qualifiers	Comments
ACCOUNT	String	EQL, NEQ	Valid account name or list of names. The asterisk (*) and the percent sign (%) wildcard characters are allowed.
AUTHPRI	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	On Alpha, valid authorized base priority (0–63). On VAX, valid authorized base priority (0–31).
CANCEL			Cancels the selection criteria for this context.
CURPRIV	Keyword	ALL, ANY, EQL, NEQ	Valid privilege name keyword or list of keywords. For more information, see the *OpenVMS Guide to System Security*.
GRP	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	UIC group number.
HW_MODEL	Integer	EQL, NEQ	Valid hardware model number.
HW_NAME	String	EQL, NEQ	Valid hardware name or a list of keywords. The asterisk (*) and the percent sign (%) wildcard characters are allowed.
JOBPRCCNT	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Subprocess count for entire job.
JOBTYPE	Keyword	EQL, NEQ	Valid job-type keyword. Valid keywords are DETACHED, NETWORK, BATCH, LOCAL, DIALUP, and REMOTE. For more information, see the *OpenVMS User's Manual*.
MASTER_PID	String	EQL, NEQ	PID of master process.
MEM	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	UIC member number.
MODE	Keyword	EQL, NEQ	Valid process mode keyword. Valid keywords are OTHER, NETWORK, BATCH, and INTERACTIVE. For more information, see the *OpenVMS User's Manual*.
NODE_CSID	Integer	EQL, NEQ	Node's cluster ID number.

Selection Item	Selection Value	Value Qualifiers	Comments
NODENAME	String	EQL, NEQ	Node name or list of node names. The asterisk (*) and the percent sign (%) wildcard characters are allowed. The default is your local node. To request all nodes, use the value ''*''.
OWNER	String	EQL, NEQ	PID of immediate parent process.
PRCCNT	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Subprocess count of process.
PRCNAM	String	EQL, NEQ	Process name or list of process names. The asterisk (*) and the percent sign (%) wildcard characters are allowed.
PRI	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Process priority level number (0–63, on Alpha), (0–31, on VAX).
PRIB	Integer	GEQ, GTR, LEQ, LSS, EQL, NEQ	Base process priority level number (0–63, on Alpha), (0–31, on VAX).
STATE	Keyword	EQL, NEQ	Valid process state keyword. For more information, see the description of the $GETJPI service in the *OpenVMS System Services Reference Manual*.
STS	Keyword	EQL, NEQ	Valid process status keyword. For more information, see the description of the $GETJPI service in the *OpenVMS System Services Reference Manual*.
TERMINAL	String	EQL, NEQ	Terminal name or list of names. The asterisk (*) and the percent sign (%) wildcard characters are allowed.
UIC	String	EQL, NEQ	User identification code (UIC) identifier (that is, of the form ''[group,member]'').
USERNAME	String	EQL, NEQ	User name or list of user names. The asterisk (*) and the percent sign (%) wildcard characters are allowed.

selection-value

Specifies the value of the selection criteria. For example, to process all the processes running on node MYVAX, specify ''MYVAX'' with the ''NODENAME'' keyword. For example:

```
$ X = F$CONTEXT("PROCESS", ctx, "NODENAME", "MYVAX", "EQL")
```

Values that are lists are valid with some selection items. If you specify more than one item, separate them with commas (,). The following example specifies a list of the nodes MYVAX, HERVAX, and HISVAX:

```
$ X=F$CONTEXT("PROCESS",ctx,"NODENAME","MYVAX,HERVAX,HISVAX","EQL")
```

You can use the asterisk (*) and the percent sign (%) wildcard characters for some values. Using wildcard characters for selection items is similar to using wildcard characters for file names.

value-qualifier
Specifies qualifiers for selection values. You must qualify selection values.

You can qualify a number, for example, by requesting that the selection be based on one of the following process values:

- LSS — less than the value specified in the call to F$PID
- LEQ — less than or equal to the value specified in the call to F$PID
- GTR — greater than the value specified in the call to F$PID
- GEQ — greater than or equal to the value specified in the call to F$PID
- EQL — equal to the value specified in the call to F$PID
- NEQ — not equal to the value specified in the call to F$PID

You can qualify some lists with the ALL, ANY, EQL, or NEQ keywords. Such lists are usually masks such as the process privilege mask, which consists of the set of enabled privileges.

- ALL — requires that all items in the list be true for a process
- ANY — requests that any item in the list be part of the attributes of a process
- EQL — requires the values to match exactly (that is, values not specified must not be true of the process)
- NEQ — requires that the value must not match

When using multiple selection values with a particular selection qualifier, a match on any one of the selection criteria is considered valid (as if an OR operand was in place); the selection values are not cumulative criteria (as if an AND operand was in place).

The difference between ALL and EQL is that the values specified with ALL must exist, but other unspecified values can exist also. EQL requires that all values specified must exist, and all others may not. For example, to request those processes whose current privileges include TMPMBX (temporary mailbox) and OPER (operator), but may include other privileges, specify the ALL keyword. To request those processes whose current privileges are TMPMBX and OPER exclusively, specify the EQL keyword.

Description

Use the F$CONTEXT function to set up selection criteria for the F$PID function.

The F$CONTEXT function is called as many times as necessary to produce the criteria needed; however, each call can specify only one selection item. Lists of item values are allowed, where appropriate, and more than one context can be operated upon at a time.

After establishing the selection criteria with appropriate calls to F$CONTEXT, F$PID is called repeatedly to return all the process identification (PID) numbers that meet the criteria specified in the F$CONTEXT function. When there are no more such processes, the F$PID function returns a null string.

After the F$PID function is called, the context symbol is considered ‘‘frozen’’; F$CONTEXT cannot be called again with the same context symbol until the associated context selection criteria have been deleted. If you attempt to set up additional selection criteria with the same context symbol, an error message is displayed. However, the context and selection criteria are not affected and calls to the F$PID function can continue.

The F$CONTEXT function uses process memory to store the selection criteria. This memory is deleted under two circumstances. Memory is deleted when the F$PID function is called and a null string ("") is returned—that is, when all processes that meet the selection criteria have been returned. Memory also is deleted if the CANCEL selection-item keyword is used in a call to F$CONTEXT with an established context. This type of call is appropriate for a Ctrl/Y operation or another condition handling routine.

Examples

1.
```
$!Establish an error and Ctrl/Y handler
$!
$ ON ERROR THEN GOTO error
$ ON CONTROL_Y THEN GOTO error
$!
$ ctx = ""
$ temp = F$CONTEXT ("PROCESS", ctx, "NODENAME", "*","EQL")
$ temp = F$CONTEXT ("PROCESS", ctx, "USERNAME", "M*,SYSTEM","EQL")
$ temp = F$CONTEXT ("PROCESS", ctx, "CURPRIV", "SYSPRV,OPER", "ALL")
$!
$!Loop over all processes that meet the selection criteria.
$!Print the PID and the name of the image for each process.
$!
$loop:
$ pid = F$PID(ctx)
$ IF pid .EQS. ""
$ THEN
$     GOTO endloop
$ ELSE
$     image = F$GETJPI(pid,"IMAGNAME")
$     SHOW SYMBOL pid
$     WRITE SYS$OUTPUT image
$     GOTO loop
$ ENDIF
$!The loop over the processes has ended.
$!
$endloop:
$!
$ EXIT
$!
$!Error handler. Clean up the context's memory with
$!the CANCEL selection item keyword.
$!
$error:
$ IF F$TYPE(ctx) .eqs. "PROCESS_CONTEXT" THEN -
_$ temp = F$CONTEXT ("PROCESS", ctx, "CANCEL")
$!
$ EXIT
```

In this example, F$CONTEXT is called three times to set up selection criteria.

The first call requests that the search take place on all nodes in the cluster. The second call requests that only the processes whose user name either starts with an ''M'' or is ''SYSTEM'' be processed. The third call restricts the selection to those processes whose current privileges include both SYSPRV (system privilege) and OPER (operator) and can have other privileges set.

The command lines between the labels ''loop'' and ''endloop'' continually call F$PID to obtain the processes that meet the criteria set up in the F$CONTEXT calls. After retrieving each PID, F$GETJPI is called to return the name of the image running in the process. Finally, the procedure displays the name of the image.

In case of error or a Ctrl/Y operation, control is passed to *error* and the context is closed if necessary. In this example, note the check for the symbol type PROCESS_CONTEXT. If the symbol has this type, selection criteria must be canceled by a call to F$CONTEXT. If the symbol is not of the type PROCESS_CONTEXT, either selection criteria have not been set up yet in F$CONTEXT, or the symbol was used with F$PID until an error occurred or until the end of the process list was reached.

2.
```
f$context("process",ctx,"prcnam  ","symbiont*,mcote*","eql")

f$context("process",ctx,"prcnam  ","symbiont*,mcote* ","neq")

f$context("process",ctx,"prcnam  ","mcote* ","neq")
f$context("process",ctx,"prcnam  ","symbiont*","neq")
```

This example shows three sets of lexicals showing the difference between the EQL and the NEQ selection values. The first lexical function (with EQL) passes back all processes with symbiont and mcote in the process name. The second and third lexical functions (with NEQ) are equivalent in that they both will pass back all processes (processes that do not have symbiont in the process name, or processes that do not have mcote in the process name.)

F$CSID

Returns an identification number from a VMScluster system and updates the context symbol to point to the current position in the system's cluster node list.

Format

F$CSID(context-symbol)

Return Value

A character string containing the system cluster identification number in the system's list of clustered nodes. If the current system is not a member of a cluster, the first return value is null. After the last system cluster identification number is returned, the F$CSID function returns a null string ("").

Arguments

context-symbol
Specifies a symbol that DCL uses to store a pointer into the system's list of clustered nodes. The F$CSID function uses this pointer to return a cluster identification number.

Specify the **context-symbol** argument by using a symbol. The first time you use the F$CSID function, use a symbol that is either undefined or equated to the null string.

If the **context-symbol** argument is undefined or equated to a null string, the F$CSID function returns the cluster identification number of the first system in the system's cluster node list. Subsequent calls to the F$CSID function will return the cluster identification number of the rest of the nodes in the cluster.

Description

The F$CSID function returns a cluster identification number, and updates the context symbol to point to the current position in the system's cluster node list.

If the current system is not a member of a cluster, the first return value is null.

You can use the F$CSID function to obtain all of the cluster identification numbers on the system. For each cluster identification returned, the F$GETSYI function can be used to obtain information about the particular system.

Once the **context-symbol** argument is initialized by the first call, each subsequent F$CSID function call returns the cluster identification number of another node in the cluster. (Note that the cluster identification numbers are returned in random order.) After the cluster identification number of the last system in the list is returned, the F$CSID function returns a null string.

Example

```
$ IF F$GETSYI("CLUSTER_MEMBER") .EQS. "FALSE" THEN GOTO NOT_CLUSTER
$ CONTEXT = ""
$START:
$   id = F$CSID (CONTEXT)
$   IF id .EQS. "" THEN EXIT
$   nodename = F$GETSYI ("NODENAME",,id)
$   WRITE SYS$OUTPUT nodename
$   GOTO start
$NOT_CLUSTER:
$ WRITE SYS$OUTPUT "Not a member of a cluster."
$ EXIT
```

This command procedure uses the F$CSID function to display a list of cluster system names. The assignment statement declares the symbol CONTEXT, which is used as the **context-symbol** argument for the F$CSID function. Because CONTEXT is equated to a null string, the F$CSID function will return the first cluster identification number in the cluster node list.

If the F$CSID function returns a null value, then the command procedure either is at the end of the list, or is attempting this operation on a non-clustered node. The call to F$GETSYI checks whether the current node is a member of a cluster. The command procedure will exit on this condition.

If the F$CSID function does not return a null value, then the command procedure uses the identification number as the third argument to the F$GETSYI function to obtain the name of the system. The name is then displayed using the WRITE command.

F$CVSI

Converts the specified bits in the specified character string to a signed number.

Format

F$CVSI(start-bit,number-of-bits,string)

Return Value

The integer equivalent of the extracted bit field, converted as a signed value.

Arguments

start-bit

Specifies the offset of the first bit to be extracted. The low-order (rightmost) bit of a string is position number 0 for determining the offset. Specify the offset as an integer expression.

If you specify an expression with a negative value, or with a value that exceeds the number of bits in the string, then DCL displays the INVRANGE error message.

number-of-bits

Specifies the length of the bit string to be extracted, which must be less than or equal to the number of bits in the string.

If you specify an expression with a negative value, or with a value that is invalid when added to the bit position offset, then DCL displays the INVRANGE error message.

string

Specifies the string from which the bits are taken. Specify the string as a character string expression.

Examples

1.
```
$ A[0,32] = %X2B
$ SHOW SYMBOL A
  A = "+..."
$ X = F$CVSI(0,4,A)
$ SHOW SYMBOL X
  X = -5   Hex = FFFFFFFB  Octal = 37777777773
```

This example uses an arithmetic overlay to assign the hexadecimal value 2B to all 32 bits of the symbol A. For more information on arithmetic overlays, see the description of the assignment statement (=).

The symbol A has a string value after the overlay because it was previously undefined. (If a symbol is undefined, it has a string value as a result of an arithmetic overlay. If a symbol was previously defined, it retains the same data type after the overlay.) The hexadecimal value 2B corresponds to the ASCII value of the plus sign (+).

Next, the F$CVSI function extracts the low-order 4 bits from the symbol A; the low-order 4 bits contain the binary representation of the hexadecimal value B. These bits are converted, as a signed value, to an integer. The converted value, –5, is assigned to the symbol X.

2.
```
$ SYM[0,32] = %X2A
$ SHOW SYMBOL SYM
  SYM = "*..."
$ Y = F$CVSI(0,33,SYM)
%DCL-W-INVRANGE, field specification is out of bounds - check sign and size
$ SHOW SYMBOL Y
%DCL-W-UNDSYM, undefined symbol - check spelling
```

In this example, the width argument specified with the F$CVSI function is too large. Therefore, DCL issues an error message and the symbol Y is not assigned a value.

F$CVTIME

Converts absolute or a combination time string to a string of the form *yyyy-mm-dd hh:mm:ss.cc*. The F$CVTIME function can also return information about absolute, combination, or delta time string.

Format

F$CVTIME([input_time] [,output_time_format] [,output_field])

Return Value

A character string containing the requested information.

Arguments

input_time

Specifies a string containing absolute, a delta, or a combination time, or TODAY, TOMORROW, or YESTERDAY. Specify the input time string as a character string expression. If the **input_time** argument is omitted or is specified as a null string (""), the current system date and time, in absolute format, is used. If parts of the date field are omitted, the missing values default to the current date. If parts of the time field are omitted, the missing values default to zero.

For more information on these time formats, see the *OpenVMS User's Manual*.

If the **input_time** argument is a delta time, you must specify the **output_time_format** argument as DELTA.

output_time_format

Specifies the time format for the information you want returned. Specify the **output_time_format** argument as one of the following character string expressions:

ABSOLUTE	The requested information should be returned in absolute time format, which is *dd-mmm-yyyy hh:mm:ss.cc*. Single-digit days are returned with no leading space or zero.
COMPARISON (default)	The requested information should be returned in the form *yyyy-mm-dd hh:mm:ss.cc* (used for comparing two times).
DELTA	The requested information should be returned in delta format, which is *dddd-hh:mm:ss.cc*. If you specify DELTA as the **output_time_format** argument, then you must also provide a delta time specification for the **input_time** argument.

output_field

Specifies a character string expression containing one of the following (do not abbreviate): DATE, MONTH, DATETIME (default), SECOND, DAY, TIME, HOUR, WEEKDAY, HUNDREDTH, YEAR, MINUTE. The information is returned in the time format specified by the **output_time_format** argument. If the **input_time** argument is a delta time and the **output_time_format** argument is DELTA, you cannot specify MONTH, WEEKDAY, or YEAR.

When the weekday is returned, the first letter is in uppercase, and the following letters are in lowercase.

Description

When using the F$CVTIME function, you can omit optional arguments that can be used to the right of the last argument you specify. However, you must include commas (,) as placeholders if you omit optional arguments to the left of the last argument you specify.

When specifying the input time argument in either absolute or combination time format, you can specify ABSOLUTE or COMPARISON as the **output_time_format** argument; you cannot specify DELTA.

When specifying the **input_time** argument in delta time format, you must specify DELTA as the **output_time_format** argument.

Examples

1.
```
$ TIME = F$TIME()
$ SHOW SYMBOL TIME
  TIME = "14-DEC-1994 10:56:23.10"
$ TIME = F$CVTIME(TIME)
$ SHOW SYMBOL TIME
  TIME = "1994-12-14 10:56:23.10"
```

This example uses the F$TIME function to return the system time as a character string and to assign the time to the symbol TIME. Then the F$CVTIME function is used to convert the system time to an alternate time format. Note that you do not need to place quotation marks (‘‘ ’’) around the argument TIME because it is a symbol. Symbols are automatically evaluated when they are used as arguments for lexical functions.

You can use the resultant string to compare two dates (using .LTS. and .GTS. operators). For example, you can use F$CVTIME to convert two time strings and store the results in the symbols TIME_1 and TIME_2. You can compare the two values, and branch to a label, based on the following results:

```
$ IF TIME_1 .LTS. TIME_2 THEN GOTO FIRST
```

2.
```
$ NEXT = F$CVTIME("TOMORROW",,"WEEKDAY")
$ SHOW SYMBOL NEXT
 NEXT = "Tuesday"
```

In this example, the F$CVTIME returns the weekday that corresponds to the absolute time keyword ‘‘TOMORROW’’. You must enclose the arguments ‘‘TOMORROW’’ and ‘‘WEEKDAY’’ in quotation marks because they are character string expressions. Also, you must include a comma as a placeholder for the **output_time_format** argument that is omitted.

F$CVUI

Extracts bit fields from character string data and converts the result to an unsigned number.

Format

F$CVUI(start-bit,number-of-bits,string)

Return Value

The integer equivalent of the extracted bit field, converted as an unsigned value.

Arguments

start-bit

Specifies the offset of the first bit to be extracted. The low-order (rightmost) bit of a string is position number 0 for determining the offset. Specify the offset as an integer expression.

If you specify an expression with a negative value, or with a value that exceeds the number of bits in the string, DCL displays the INVRANGE error message.

number-of-bits

Specifies the length of the bit string to be extracted, which must be less than or equal to the number of bits in the string argument.

If you specify an expression with a negative value, or with a value that is invalid when added to the bit position offset, DCL displays the INVRANGE error message.

string

Specifies the character string to be edited.

Example

```
$ A[0,32] = %X2B
$ SHOW SYMBOL A
  A = "+..."
$ X = F$CVUI(0,4,A)
$ SHOW SYMBOL X
  X = 11   Hex = 0000000B  Octal = 00000000013
```

This example uses an arithmetic overlay to assign the hexadecimal value 2B to all 32 bits of the symbol A. The symbol A has a string value after the overlay because it was previously undefined. (If a symbol is undefined, it has a string value as a result of an arithmetic overlay. If a symbol was previously defined, it retains the same data type after the overlay.) The hexadecimal value 2B corresponds to the ASCII character ‘‘+’’.

Next, the F$CVUI function extracts the low-order 4 bits from the symbol A; the low-order 4 bits contain the binary representation of the hexadecimal value B. These bits are converted, as a signed value, to an integer. The converted value, 11, is assigned to the symbol X.

F$DEVICE

Returns the device names of all devices on a system that meet the specified selection criteria.

Note that the device names are returned in random order.

Format

F$DEVICE([search_devnam],[devclass],[devtype], [stream-id])

Return Value

A character string containing the name of a device in the system's list of devices. After the last device name in the system's device list is returned, the F$DEVICE function returns a null string ("").

Arguments

search_devnam

Specifies the name of the device for which F$DEVICE is to search. The asterisk (*) and the percent sign (%) wildcard characters are allowed in the **search_devnam** argument.

Specify the **search_devnam** argument as a character string expression.

devclass

Specifies the device class for which F$DEVICE is to search. Specify the **devclass** argument as a character string expression that corresponds to a valid device class name. See Table DCLI–7 under F$GETDVI.

devtype

Specifies the device type for which F$DEVICE is to search. Specify the **devtype** argument as a character string expression that corresponds to a valid type name. Valid type names can be found in Table DCLI–8 under F$GETDVI.

Note

Specifying a device type without specifying a device class will result in an error.

stream-id

A positive integer representing the search stream identification number.

The search stream identification number is used to maintain separate search contexts when you use the F$DEVICE function more than once and when you supply different search criteria. If you use the F$DEVICE function more than once in a command procedure and if you also use different search criteria, specify **stream-id** arguments to identify each search separately.

If the search criteria are changed in a call before the device name list is exhausted, the context will be reinitialized and the search will restart.

If you omit the **stream-id** argument, the F$DEVICE function assumes an implicit single search stream. That is, the F$DEVICE function starts searching at the beginning each time you specify different search criteria.

Description

The F$DEVICE function allows you to search for devices that meet certain search criteria by using the $DEVICE_SCAN system service.

The F$DEVICE function allows asterisk (*) and percent sign (%) wildcard character searches based only on the device name; you must specify a valid character string expression for the device class or device type.

You can use the F$DEVICE function in a loop in a command procedure to return device names that match the specified selection criteria. Each time the F$DEVICE function is executed, it returns the next device on the system that matches the selection criteria. Note that devices are returned in no particular order. After the last device name is returned, the next F$DEVICE function returns a null string.

Note that you must maintain the context of the search string explicitly (by specifying the **stream-id** argument) or implicitly (by omitting the **stream-id** argument). In either case, you must specify the same selection criteria each time you execute the F$DEVICE system service with the same (explicit or implicit) stream.

Example

```
$ START:
$     DEVICE_NAME = F$DEVICE("*0:","DISK","RA60")
$     IF DEVICE_NAME .EQS. "" THEN EXIT
$     SHOW SYMBOL DEVICE_NAME
$     GOTO START
```

This command procedure displays the device names of all the RA60s on a unit numbered 0.

Because no **stream-id** argument is specified, F$DEVICE uses an implicit search stream. Each subsequent use of the F$DEVICE function uses the same search criteria to return the next device name. After the last device name is displayed, the F$DEVICE function returns a null string and the procedure exits.

F$DIRECTORY

Returns the current default directory name string. The F$DIRECTORY function has no arguments, but must be followed by parentheses.

Format

F$DIRECTORY( )

Return Value

A character string for the current default directory name, including brackets ([ ]). If you use the SET DEFAULT command and specify angle brackets (<>) in a directory specification, the F$DIRECTORY function returns angle brackets in the directory string.

Arguments

None.

Description

You can use the F$DIRECTORY function to save the name of the current default directory in a command procedure, to change the default to another directory to do work, and to later restore the original setting.

Example

```
$ SAVE_DIR = F$DIRECTORY()
$ SET DEFAULT [MALCOLM.TESTFILES]
   .
   .
   .
$ SET DEFAULT 'SAVE_DIR'
```

This example shows an excerpt from a command procedure that uses the F$DIRECTORY function to save the current default directory setting. The assignment statement equates the symbol SAVE_DIR to the current directory. Then the SET DEFAULT command establishes a new default directory. Later, the symbol SAVE_DIR is used in the SET DEFAULT command that restores the original default directory.

Note that you can use the F$ENVIRONMENT function with the DEFAULT keyword to return the default disk and directory. You should use the F$ENVIRONMENT function rather than the F$DIRECTORY function in situations involving more than one disk.

F$EDIT

Edits the character string based on the edits specified in the **edit-list** argument.

Format

F$EDIT(string, edit-list)

Return Value

A character string containing the specified edits.

Arguments

string
Specifies a character string to be edited. Quoted sections of the string are not edited.

edit-list
Specifies a character string containing one or more of the following keywords which specify the types of edits to be made to the string:

Edit	Action
COLLAPSE	Removes all spaces or tabs.
COMPRESS	Replaces multiple spaces or tabs with a single space.
LOWERCASE	Changes all uppercase characters to lowercase.
TRIM	Removes leading and trailing spaces or tabs.
UNCOMMENT	Removes comments.
UPCASE	Changes all lowercase characters to uppercase.

If you specify more than one keyword, separate them with commas (,). Do not abbreviate these keywords.

Edits are not applied to quoted sections of strings. Therefore, if a string contains quotation marks (‘‘ ’’), the characters within the quotation marks are not affected by the edits specified in the edit list.

Note

When UPCASE is specified with LOWERCASE in an edit-list, UPCASE takes precedence.

Examples

1.
```
$ LINE = "   THIS   LINE   CONTAINS A ""  QUOTED  "" WORD"
$ SHOW SYMBOL LINE
  LINE =  "    THIS    LINE    CONTAINS A "  QUOTED  " WORD"
$ NEW_LINE = F$EDIT(LINE, "COMPRESS, TRIM")
$ SHOW SYMBOL NEW_LINE
  NEW_LINE = "THIS LINE CONTAINS A "  QUOTED  " WORD"
```

This example uses the F$EDIT function to compress and trim a string by replacing multiple blanks with a single blank, and by removing leading and trailing blanks. The string LINE contains quotation marks around the word

QUOTED. (To enter quotation marks into a character string, use double quotation marks in the assignment statement.)

Note that the F$EDIT function does not compress the spaces in the quoted section of the string; therefore, the spaces are retained around the word QUOTED.

2.
```
$ LOOP:
$       READ/END_OF_FILE = DONE INPUT_FILE RECORD
$       RECORD = F$EDIT(RECORD, "TRIM, UPCASE")
$       WRITE OUTPUT_FILE RECORD
$       GOTO LOOP
    .
    .
    .
```

This example sets up a loop to read records from a file, to edit them, and to write them to an output file. The edited records have leading and trailing blanks removed, and are converted to uppercase.

3.
```
$ UNCOMMENT_LINE = F$EDIT("$ DIR ! THIS IS THE COMMENT", "UNCOMMENT")
$ SHOW SYMBOL UNCOMMENT_LINE
$ UNCOMMENT_LINE = "$ DIR"
```

This example uses the F$EDIT function to remove comments.

F$ELEMENT

Extracts one element from a string of elements.

Format

F$ELEMENT(element-number, delimiter, string)

Return Value

A character string containing the specified element.

Arguments

element-number
Specifies the number of the element to extract (numbering begins with zero). Specify the **element-number** argument as an integer expression. If the **element-number** argument exceeds the number of elements in the string, F$ELEMENT returns the delimiter.

delimiter
Specifies a character used to separate the elements in the string. Specify the delimiter as a character string expression.

string
Specifies a string containing a delimited list of elements. Specify the string as a character string expression.

Examples

1.
```
$ DAY_LIST = "MON/TUE/WED/THU/FRI/SAT/SUN"
$ INQUIRE DAY "ENTER DAY (MON TUE WED THU FRI SAT SUN)"
$ NUM = 0
$ LOOP:
$       LABEL = F$ELEMENT(NUM,"/",DAY_LIST)
$       IF LABEL .EQS. "/" THEN GOTO END
$       IF DAY .EQS. LABEL THEN GOTO 'LABEL'
$       NUM = NUM +1
$       GOTO LOOP
$
$ MON:
    .
    .
    .
```

This example sets up a loop to test an input value against the elements in a list of values. If the value for DAY matches one of the elements in DAY_LIST, control is passed to the corresponding label. If the value returned by the F$ELEMENT function matches the delimiter, the value DAY was not present in the DAY_LIST, and control is passed to the label END.

2.
```
$ ! INDEX.COM
$ !
$ CHAPTERS = "0,1,2,3,4,5,6,A,B,C"
$ NEXT = 0
$ LOOP:
$   NEXT = NEXT + 1
$   NUM = F$ELEMENT(NEXT,",",CHAPTERS)
$   IF (NUM .NES. ",")
$   THEN
$      RUN INDEX CHAP'NUM'
$      GOTO LOOP
$   ENDIF
```

This example processes files named CHAP1, CHAP2, ... CHAP6, CHAPA, CHAPB, and CHAPC, in that order. (Zero is included in the CHAPTERS string to initialize the procedure logic.) NEXT is initialized to zero. The procedure enters the loop. In the first iteration, NEXT is incremented to 1 and the result of the F$ELEMENT call is the string ''1''. The procedure runs the index, chapter 1. In the second iteration, NEXT is incremented to 2 and the result of the F$ELEMENT call is the string ''1''. The procedure runs the index, chapter 2. Processing continues until the result of the F$ELEMENT call is the delimiter specified in the call.

F$ENVIRONMENT

Returns information about the current DCL command environment.

Format

F$ENVIRONMENT(item)

Return Value

Information that corresponds to the specified item. The return value can be either an integer or a character string, depending on the specified item.

Argument

item
Specifies the type of information to be returned. Specify one of the following keywords (do not abbreviate these keywords):

Item	Data Type	Information Returned
CAPTIVE	String	TRUE if you are logged in to a captive account. The system manager can define captive accounts in the user authorization file (UAF) by using the Authorize utility (AUTHORIZE).
CONTROL	String	Control characters currently enabled with SET CONTROL. Multiple characters are separated by commas; if no control characters are enabled, the null string ("") is returned.
DEFAULT	String	Current default device and directory name. The returned string is the same as SHOW DEFAULT output.
DEPTH	Integer	Current command procedure depth. The command procedure depth is 0 when you log in interactively and when you submit a batch job. The command procedure depth is 1 when you execute a command procedure interactively or from within a batch job. A nested command procedure has a depth of 1 greater than the depth of the command procedure from which the nested procedure is executed.
DISIMAGE	String	TRUE if you are logged in to an account that does not allow the use of the RUN and MCR commands or foreign commands. The system manager can add or remove the DISIMAGE attribute for accounts in the UAF by using AUTHORIZE.
INTERACTIVE	String	TRUE if the process is executing interactively.

Item	Data Type	Information Returned
KEY_STATE	String	Current locked keypad state. See the description of the DEFINE/KEY command for more information on keypad states.
MAX_DEPTH	Integer	Maximum allowable command procedure depth.
MESSAGE	String	Current setting of SET MESSAGE qualifiers. Each qualifier in the string is prefaced by a slash (/); therefore, the output from F$ENVIRONMENT(''MESSAGE'') can be appended to the SET MESSAGE command to form a valid DCL command line.
NOCONTROL	String	Control characters currently disabled with SET NOCONTROL. Multiple characters are separated by commas (,); if no control characters are disabled, the null string is returned.
ON_CONTROL_Y	String	If issued from a command procedure, returns TRUE if ON_CONTROL_Y is set. ON_CONTROL_Y always returns FALSE at DCL command level.
ON_SEVERITY	String	If issued from a command procedure, returns the severity level at which the action specified with the ON command is performed. ON_SEVERITY returns NONE when SET NOON is in effect or at DCL command level.
OUTPUT_RATE	String	Delta time string containing the default output rate, which indicates how often data is written to the batch job log file while the batch job is executing. OUTPUT_RATE returns a null string if used interactively.
PROCEDURE	String	File specification of the current command procedure. If used interactively, the terminal device name is returned.
PROMPT	String	Current DCL prompt.
PROMPT_CONTROL	String	TRUE if a carriage return and line feed precede the prompt.
PROTECTION	String	Current default file protection. The string can be used with the SET PROTECTION /DEFAULT command to form a valid DCL command line.
RESTRICTED	String	TRUE if you are logged in to a restricted account. The system manager can define restricted accounts in the UAF by using AUTHORIZE.

Item	Data Type	Information Returned
SYMBOL_SCOPE	String	[NO]LOCAL, [NO]GLOBAL to indicate the current symbol scoping state.
VERB_SCOPE	String	[NO]LOCAL, [NO]GLOBAL to indicate the current symbol scoping state for verbs. (For more information, see the description of the SET SYMBOL command.)
VERIFY_IMAGE	String	TRUE if image verification (SET VERIFY=IMAGE) is in effect. If image verification is in effect, then the command procedure echoes input data read by images.
VERIFY_PREFIX	String	Returns the prefix control string set by means of the SET PREFIX command.
VERIFY_ PROCEDURE	String	TRUE if procedure verification SET VERIFY=PROCEDURE is in effect. If command verification is in effect, then the command procedure echoes DCL command lines.

Examples

1.
```
$ SAVE_MESSAGE = F$ENVIRONMENT("MESSAGE")
$ SET MESSAGE/NOFACILITY/NOIDENTIFICATION
    .
    .
    .
$ SET MESSAGE'SAVE_MESSAGE'
```

This example uses the F$ENVIRONMENT function to save the current message setting before changing the setting. At the end of the command procedure, the original message setting is restored. The single quotation marks (‘ ’) surrounding the symbol SAVE_MESSAGE indicate that the value for the symbol should be substituted.

2.
```
$ MAX = F$ENVIRONMENT("MAX_DEPTH")
$ SHOW SYMBOL MAX
  MAX = 32   Hex = 00000020  Octal = 00000000040
```

This example uses the F$ENVIRONMENT function to determine the maximum depth allowable within command procedures.

3.
```
$ SAVE_PROT = F$ENVIRONMENT("PROTECTION")
$ SET PROTECTION = (SYSTEM:RWED, OWNER:RWED, GROUP, WORLD)/DEFAULT
    .
    .
    .
$ SET PROTECTION = ('SAVE_PROT')/DEFAULT
```

This example uses the F$ENVIRONMENT function to save the current default protection before changing the protection. At the end of the command procedure, the original protection is restored. You must place single quotation marks around the symbol SAVE_PROT to request symbol substitution.

F$EXTRACT

Extracts the specified characters from the specified string.

Format

F$EXTRACT(start,length,string)

Return Value

A character string containing the characters delimited by the **start** and **length** arguments.

Arguments

start

Specifies the offset of the starting character of the string you want to extract. Specify the start argument as an integer expression that is greater than or equal to zero.

The offset is the relative position of a character or a substring with respect to the beginning of the string. Offset positions begin with zero. The string always begins with the leftmost character.

If you specify an offset that is greater than or equal to the length of the string, F$EXTRACT returns a null string ("").

length

Specifies the number of characters you want to extract; must be less than or equal to the size of the string. Specify the length as an integer expression that is greater than or equal to zero.

If you specify a length that exceeds the number of characters from the offset to the end of the string, the F$EXTRACT function returns the characters from the offset through the end of the string.

string

Specifies the character string to be edited. Specify the string as a character string expression.

Examples

1.
```
$ NAME = "JOE SMITH"
$ FIRST = F$EXTRACT(0,3,NAME)
$ SHOW SYMBOL FIRST
  FIRST = "JOE"
```

This portion of a command procedure uses the F$EXTRACT function to extract the first 3 characters from the character string assigned to the symbol NAME. The offset and length arguments are integers, and the string argument is a symbol. You do not need to use quotation marks (‘‘ ’’) around integers or symbols when they are used as arguments for lexical functions.

2.
```
$ P1 = "MYFILE.DAT"
$ FILENAME = F$EXTRACT(0,F$LOCATE(".",P1),P1)
```

This portion of a command procedure shows how to locate a character within a string, and how to extract a substring ending at that location.

The lexical function F$LOCATE gives the numeric value representing the offset position of a period in the character string value of P1. (The offset position of the period is equal to the length of the substring before the period.)

This F$LOCATE function is used as an argument in the F$EXTRACT function to specify the number of characters to extract from the string. If a procedure is invoked with the parameter MYFILE.DAT, these statements result in the symbol FILENAME being given the value MYFILE.

Note that the F$LOCATE function in the above example assumes that the file specification does not contain a node name or a directory specification containing a subdirectory name. To obtain the file name from a full file specification, use the F$PARSE function.

3.
```
$ IF F$EXTRACT(12,2,F$TIME()) .GES. "12" THEN GOTO AFTERNOON
$ MORNING:
$ WRITE SYS$OUTPUT "Good morning!"
$ EXIT
$ AFTERNOON:
$ WRITE SYS$OUTPUT "Good afternoon!"
$ EXIT
```

This example shows a procedure that displays a different message, depending on whether the current time is morning or afternoon. It first obtains the current time of day by using the F$TIME function. The F$TIME function returns a character string, which is the string argument for the F$EXTRACT function. The F$TIME function is automatically evaluated when it is used as an argument, so you do not need to use quotation marks.

Next, the F$EXTRACT function extracts the hours from the date and time string returned by F$TIME. The string returned by F$TIME always contains the hours field beginning at an offset of 12 characters from the start of the string.

The F$EXTRACT function extracts 2 characters from the string, beginning at this offset, and compares the string value extracted with the string value 12. If the comparison is true, then the procedure writes ‘‘Good afternoon!’’. Otherwise, it writes ‘‘Good morning!’’.

Note that you can also use the F$CVTIME function to extract the hour field from a time specification. This method is easier than the one shown in the above example.

F$FAO

Converts character and numeric input to ASCII character strings. (FAO stands for formatted ASCII output.) By specifying formatting instructions, you can use the F$FAO function to convert integer values to character strings, to insert carriage returns and form feeds, to insert text, and so on.

Format

F$FAO(control-string[,argument[,...]])

Return Value

A character string containing formatted ASCII output. This output string is created from the fixed text and FAO directives in the control string.

Arguments

control-string
Specifies the fixed text of the output string, consisting of text and any number of FAO directives. The control string may be any length. Specify the control string as a character string expression.

The F$FAO function uses FAO directives to modify or insert ASCII data into the fixed text in the control string.

Table DCLI–4 lists the FAO directives you can specify in a control string.

argument[,...]
Specifies from 1 to 15 arguments required by the FAO directives used in the control string. Specify the arguments as integer or character string expressions. Table DCLI–4 lists the argument types required by each FAO directive.

FAO directives may require one or more arguments. The order of the arguments must correspond exactly with the order of the directives in the control string. In most cases, an error message is not displayed if you misplace an argument.

If you specify an argument whose type (integer or string) does not match that of the corresponding directive, unpredictable results are returned. You can use the F$INTEGER and F$STRING lexical functions to convert arguments to the proper type.

If there are not enough arguments listed, F$FAO continues reading past the end of an argument list. Therefore, always be sure to include enough arguments to satisfy the requirements of all the directives in a control string.

If you specify an invalid parameter for any directive, you may see unexpected errors, which indicate that the command did not succeed. (These errors are passed through to you from the $FAO system service.)

Description

The F$FAO lexical function invokes the $FAO system service to convert character and numeric input to ASCII character strings. (FAO stands for formatted ASCII output.) By specifying formatting instructions, you can use the F$FAO function to convert integer values to character strings, to insert carriage returns and form feeds, to insert text, and so on.

Specify an FAO directive using any one of the following formats:

Format	Function
!DD	One directive
!n(DD)	A directive repeated a specified number of times
!lengthDD	A directive that places its output in a field of a specified length
!n(lengthDD)	A directive that is repeated a specified number of times and generates output fields of a specified length

The exclamation point (!) indicates that the following character or characters are to be interpreted as an FAO directive. *DD* represents a 1- or 2-character uppercase code indicating the action that F$FAO is to perform. When specifying repeat counts, *n* is a decimal value specifying the number of times the directive is to be repeated. The *length* value is a decimal number that instructs F$FAO to generate an output field of ''length'' characters.

Repeat counts and output lengths may also be specified by using a number sign (#) in place of absolute numeric value. If you use a number sign, you must specify the numeric value as an integer expression in the corresponding place in the argument list.

When a variable output field is specified with a repeat count, only one length parameter is required, because each output string has the specified length.

The FAO directives are grouped in the following categories:

- Character string insertion
- Zero-filled numeric conversion
- Blank-filled numeric conversion
- Special formatting
- Parameter interpretation

Table DCLI–4 summarizes the FAO directives and shows the required argument types. In addition, the following sections describe output strings from directives that perform character string insertion, zero-filled numeric conversion, and blank-filled numeric conversion.

Note

Two types of directives that are supported by the $FAO system service are not supported by the DCL F$FAO lexical function. These types are:

- Quadword numeric directives, which are not supported in DCL because all DCL numeric values are stored and manipulated as longwords.
- String directives other than the !AS directive, which are not supported in DCL because all DCL strings are stored and manipulated by descriptor.

For further information on the $FAO system service directive, see the *OpenVMS System Services Reference Manual*.

Table DCLI–4 Summary of FAO Directives

Directive	Argument Type	Description
Character string insertion:		
!AS	String	Inserts a character string as is.
Zero-filled numeric conversion:		
!OB	Integer	Converts a byte to octal notation.
!OW	Integer	Converts a word to octal notation.
!OL	Integer	Converts a longword to octal notation.
!XB	Integer	Converts a byte to hexadecimal notation.
!XW	Integer	Converts a word to hexadecimal notation.
!XL	Integer	Converts a longword to hexadecimal notation.
!ZB	Integer	Converts a byte to decimal notation.
!ZW	Integer	Converts a word to decimal notation.
!ZL	Integer	Converts a longword to decimal notation.
Blank-filled numeric conversion:		
!UB	Integer	Converts a byte to decimal notation without adjusting for negative numbers.
!UW	Integer	Converts a word to decimal notation without adjusting for negative numbers.
!UL	Integer	Converts a longword to decimal notation without adjusting for negative numbers.
!SB	Integer	Converts a byte to decimal notation with negative numbers converted properly.
!SW	Integer	Converts a word to decimal notation with negative numbers converted properly.
!SL	Integer	Converts a longword to decimal notation with negative numbers converted properly.
Special formatting:		
!/	None	Inserts a carriage return and a line feed.
!_	None	Inserts a tab.
!^	None	Inserts a form feed.
!!	None	Inserts an exclamation point (!).
!%I	Integer	Converts a longword integer to a named UIC in the format [group-identifier,member-identifier].

(continued on next page)

Table DCLI–4 (Cont.) Summary of FAO Directives

Directive	Argument Type	Description
!%S	None	Inserts an ''s'' if the most recently converted number is not 1. (Not recommended for use with multilingual products.)
!%U	Integer	Converts a longword integer to a numeric UIC in the format [g,m], where *g* is the group number and *m* is the member number.
		The directive inserts the brackets and the comma.
!n<...!>	None	Left-justifies and blank-fills all data represented by the instructions . . . in fields *n* characters wide.
!n*c*	None	Repeats the character represented by *c* for *n* times.
!n%C	String	Inserts a character string when the most recently evaluated argument has the value *n*. (Recommended for use with multilingual products.)
!%E	String	Inserts a character string when the value of the most recently evaluated argument does not match any preceding !n%C directives. (Recommended for use with multilingual products.)
!%F	None	Marks the end of a plurals statement.
!%T	Integer equal to 0	Inserts the current time.
!%D	Integer equal to 0	Inserts the current date/time.
Argument interpretation:		
!-	None	Reuses the last argument.
!+	None	Skips the next argument.

Output Strings from Character String Insertion

The !AS directive inserts a character string (specified as an argument for the directive) into the control string. The field length of the character string when it is inserted into the control string defaults to the length of the character string. If the default length is shorter than an explicitly stated field length, the string is left-justified and blank-filled. If the default length is longer than an explicitly stated field length, the string is truncated on the right.

Output Strings from Zero-Filled Numeric Conversion

Directives for zero-filled numeric conversion convert an integer (specified as an argument for the directive) to decimal, octal, or hexadecimal notation. The ASCII representation of the integer is inserted into the control string. Default output field lengths for the converted argument are determined as follows.

Directives that convert arguments to octal notation return 3 digits for byte conversion, 6 digits for word conversion, and 11 digits for longword conversion. Numbers are right-justified and zero-filled on the left. Explicit-length fields longer than the default are blank-filled on the left. Explicit-length fields shorter than the default are truncated on the left.

Directives that convert arguments to hexadecimal notation return 2 digits for byte conversion, 4 digits for word conversion, and 8 digits for longword conversion. Numbers are right-justified and zero-filled on the left. Explicit-length fields longer than the default are blank-filled on the left. Explicit-length fields shorter than the default are truncated on the left.

Directives that convert arguments to decimal notation return the required number of characters for the decimal number. Explicit-length fields longer than the default are zero-filled on the left. If an explicit-length field is shorter than the number of characters required for the decimal number, the output field is completely filled with asterisks (*).

For byte conversion, only the low-order 8 bits of the binary representation of the argument are used. For word conversion, only the low-order 16 bits of the binary representation of the argument are used. For longword conversion, the entire 32-bit binary representation of the argument is used.

Output Strings from Blank-Filled Numeric Conversion

Directives for blank-filled numeric conversion convert an integer (specified as an argument for the directive) to decimal notation. These directives can convert the integer as a signed or unsigned number. The ASCII representation of the integer is inserted into the control string.

Output field lengths for the converted argument default to the required number of characters. Values shorter than explicit-length fields are right-justified and blank-filled; values longer than explicit-length fields cause the field to be filled with asterisks.

For byte conversion, only the low-order 8 bits of the binary representation of the argument are used. For word conversion, only the low-order 16 bits of the binary representation of the argument are used. For longword conversion, the entire 32-bit binary representation of the argument is used.

Output Strings from Special Formatting Directives

The !n%C and !%E directives insert an ASCII string (based on the value of the most recently evaluated argument) into the output string. These directives are useful for inserting irregular plural nouns and verbs.

If the most recently evaluated argument equals *n*, the text between one directive and the next is inserted into the output string. If the most recently evaluated argument does not equal *n*, the next !n%C directive is processed.

If *n* must be a negative number, you must specify it as an argument and use the number sign (#).

You can specify the !n%C and !%E directives with repeat counts. If you specify repeat counts, the text between one directive and the next is copied to the output string the specified number of times.

The %F directive marks the end of a plurals statement.

Examples

1.
```
$ COUNT = 57
$ REPORT = F$FAO("NUMBER OF FORMS = !SL",COUNT)
$ SHOW SYMBOL REPORT
  REPORT = "NUMBER OF FORMS = 57"
```

In this command procedure, the FAO directive !SL is used in a control string to convert the number equated to the symbol COUNT to a character string. The converted string is inserted into the control string.

Note that COUNT is assigned an integer value of 57. The F$FAO function returns the ASCII string, ''NUMBER OF FORMS = 57'', and assigns the string to the symbol REPORT.

2.
```
$ A = "ERR"
$ B = "IS"
$ C = "HUM"
$ D = "AN"
$ PHRASE = F$FAO("TO !3(AS)",A,B,C+D)
$ SHOW SYMBOL PHRASE
$ PHRASE = "TO ERRISHUMAN"
```

In this command procedure, the !AS directive is used to insert the values assigned to the symbols A, B, C, and D into the control string.

Because the specified repeat count for the !AS directive is 3, F$FAO looks for three arguments. The arguments in this example include the symbol A (''ERR''), the symbol B (''IS''), and the expression C+D (''HUMAN''). Note that the values of these string arguments are concatenated to form the string ''ERRISHUMAN''.

3.
```
$ A = "ERR"
$ B = "IS"
$ C = "HUMAN"
$ PHRASE = F$FAO("TO !#(#AS)",3,6,A,B,C)
$ SHOW SYMBOL PHRASE
$ PHRASE = "TO ERR   IS    HUMAN "
```

In this command procedure, the F$FAO function is used with the !AS directive to format a character string. The first number sign (#) represents the repeat count given by the first argument, 3. The second number sign represents the field size given by the second argument, 6. The next three arguments (A,B,C) provide the strings that are placed into the control string each time the !AS directive is repeated.

Each argument string is output to a field having a length of 6 characters. Because each string is less than 6 characters, each field is left-justified and padded with blank spaces. The resulting string is assigned to the symbol PHRASE.

4.
```
$ OFFSPRING = 1
$ REPORT = F$FAO-
("There !0UL!1%Cis!%Eare!%F !-!UL !-!0UL!1%Cchild!%Echildren!%F here",1)
$ SHOW SYMBOL REPORT
$ REPORT ="There is 1 child here"
```

In this command procedure, the !0UL directive evaluates the argument OFFSPRING but does not insert the value in the output string. The !n%C directive inserts the character string ''is'' into the output string because its value and the value of the argument OFFSPRING match. The directives

!-!UL evaluate the argument a second time so that the correct character string can be inserted in the proper place in the output string. The !%F directive marks the end of each plurals statement. The F$FAO function returns the ASCII string ‘‘There is 1 child here’’ and assigns the string to the symbol REPORT.

F$FILE_ATTRIBUTES

Returns attribute information for a specified file.

Format

F$FILE_ATTRIBUTES(filespec,item)

Return Value

Either an integer or a character string, depending on the item you request. Table DCLI–5 shows the data types of the values returned for each item.

Arguments

filespec
Specifies the name of the file about which you are requesting information. You must specify the file name as a character string expression.

You can specify only one file name. Wildcard characters are not allowed.

item
Indicates which attribute of the file is to be returned. The **item** argument must be specified as a character string expression, and can be any one of the OpenVMS RMS field names listed in Table DCLI–5.

Description

Use the F$FILE_ATTRIBUTES lexical function in DCL assignment statements and expressions to return file attribute information. Table DCLI–5 lists the items you can specify with the F$FILE_ATTRIBUTES function, the information returned, and the data type of this information.

Table DCLI–5 F$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
AI	String	TRUE if after-image (AI) journaling is enabled; FALSE if disabled
ALQ	Integer	Allocation quantity
BDT	String	Backup date/time
BI	String	TRUE if before-image (BI) journaling is enabled; FALSE if disabled
BKS	Integer	Bucket size
BLS	Integer	Block size
CBT	String	TRUE if contiguous-best-try; otherwise FALSE
CDT	String	Creation date/time
CTG	String	TRUE if contiguous; otherwise FALSE
DEQ	Integer	Default extension quantity
DID	String	Directory ID string

(continued on next page)

Table DCLI–5 (Cont.) F$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
DVI	String	Device name string
EDT	String	Expiration date/time
EOF	Integer	Number of blocks used
ERASE	String	TRUE if a file's contents are erased before a file is deleted; otherwise FALSE
FFB	Integer	First free byte
FID	String	File ID string
FSZ	Integer	Fixed control area size
GBC	Integer	Global buffer count
GRP	Integer	Owner group number
JOURNAL_FILE	String	TRUE if the file is a journal; otherwise FALSE
KNOWN	String	Known file; returns TRUE or FALSE to indicate whether file is installed with the Install utility (INSTALL). However, returns NOSUCHFILE if a file does not exist (for example, the file has been installed but subsequently deleted).
LOCKED	String	TRUE if a file is deaccessed-locked; otherwise FALSE
LRL	Integer	Longest record length
MBM	Integer	Owner member number
MOVE	String	TRUE if movefile operations are enabled; otherwise FALSE
MRN	Integer	Maximum record number
MRS	Integer	Maximum record size
NOA	Integer	Number of areas
NOBACKUP	String	FALSE if the file is marked for backup; TRUE if the file is marked NOBACKUP
NOK	Integer	Number of keys
ORG	String	File organization; returns SEQ, REL, IDX
PRO	String	File protection string
PVN	Integer	Prolog version number
RAT	String	Record attributes; returns CR, PRN, FTN, ""
RCK	String	TRUE if read check; otherwise FALSE
RDT	String	Revision date/time
RFM	String	Record format string; returns the values VAR, FIX, VFC, UDF, STM, STMLF, STMCR
RU	String	TRUE if recovery unit (RU) journaling is enabled; returns TRUE or FALSE

(continued on next page)

Table DCLI–5 (Cont.) F$FILE_ATTRIBUTES Items

Item	Return Type	Information Returned
RVN	Integer	Revision number
SHELVABLE	String	TRUE if the file is shelvable; otherwise FALSE
SHELVED	String	TRUE if the file is shelved; otherwise FALSE
STORED_ SEMANTICS	String	ASCII string that represents stored semantics
UIC	String	Owner user identification code (UIC) string
WCK	String	TRUE if write check; otherwise FALSE

File attributes are stored in the file header, which is created from information in OpenVMS RMS control blocks. For more information on OpenVMS RMS control blocks, see the *OpenVMS Record Management Services Reference Manual*.

Examples

1.
```
$ FILE_ORG = F$FILE_ATTRIBUTES("QUEST.DAT","ORG")
$ SHOW SYMBOL FILE_ORG
  FILE_ORG = "SEQ"
```

This example uses the F$FILE_ATTRIBUTES function to assign the value of the file organization type to the symbol FILE_ORG. The F$FILE_ ATTRIBUTES function returns the character string SEQ to show that QUEST.DAT is a sequential file.

The QUEST.DAT and ORG arguments for the F$FILE_ATTRIBUTES function are string literals and must be enclosed in quotation marks (‘‘ ’’) when used in expressions.

2.
```
$ RFM = F$FILE_ATTRIBUTES("KANSAS::USE$:[CARS]SALES.CMD","RFM")
$ SHOW SYMBOL RFM
  RFM = "VAR"
```

This example uses the F$FILE_ATTRIBUTES function to return information about a file on a remote node. The function returns the record format string VAR, indicating that records are variable length.

F$GETDVI

Returns a specified item of information for a specified device.

Format

F$GETDVI(device-name,item)

Return Value

Either an integer or a character string, depending on the item you request. Table DCLI–6 shows the data types of the values returned for each item.

Arguments

device-name
Specifies a physical device name or a logical name equated to a physical device name. Specify the device name as a character string expression.

After the **device-name** argument is evaluated, the F$GETDVI function examines the first character of the name. If the first character is an underscore (_), the name is considered a physical device name. Otherwise, a single level of logical name translation is performed and the equivalence name, if any, is used.

item
Specifies the type of device information to be returned. The **item** argument must be specified as a character string expression and can be any one of the items listed in Table DCLI–6.

Description

The F$GETDVI lexical function invokes the $GETDVI system service to return a specified item of information for a specified device. You can obtain a list of devices on your current system by using the lexical function F$DEVICE. Unless otherwise stated in the description of the item argument, F$GETDVI returns device information about the local node only.

This lexical function allows a process to obtain information for a device to which the process has not necessarily assigned a channel.

The F$GETDVI function returns information on all items that can be specified with the $GETDVI system service. In addition to the items that the $GETDVI system service allows, the F$GETDVI function allows you to specify the item EXISTS.

Table DCLI–6 lists the items you can specify with the F$GETDVI function, the type of information returned, and the data types of the return values. In addition to the return information listed in Table DCLI–6, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI. Table DCLI–7 lists the values returned by the DEVCLASS item. Table DCLI–8 lists the values returned by the DEVTYPE item. For more information on the $GETDVI system service and the items you can specify, see the *OpenVMS System Services Reference Manual*.

Table DCLI–6 F$GETDVI Items

Item	Return Type	Information Returned[1]
ACPPID	String	Ancillary control process (ACP) identification.
ACPTYPE	String	ACP type code, as one of the following strings: F11V1, F11V2, JNL, MTA, NET, REM, or ILLEGAL if the device is not mounted or is mounted using the /FOREIGN qualifier.
ALL	String	TRUE or FALSE to indicate whether the device is allocated.
ALLDEVNAM	String	Allocation class device name.
ALLOCLASS	Longword integer between 0 and 255	Allocation class of the host.
ALT_HOST_AVAIL	String	TRUE or FALSE to indicate whether the host serving the alternate path is available.
ALT_HOST_NAME	String	Name of the host serving the alternate path.
ALT_HOST_TYPE	String	Hardware type of the host serving the alternate path.
AVL	String	TRUE or FALSE to indicate whether the device is available for use.
CCL	String	TRUE or FALSE to indicate whether the device is a carriage control device.
CLUSTER	Integer	Volume cluster size.
CONCEALED	String	TRUE or FALSE to indicate whether the logical device name translates to a concealed device.
CYLINDERS	Integer	Number of cylinders on the volume (disks only).
DEVBUFSIZ	Integer	Device buffer size.
DEVCHAR	Integer	Device characteristics.
DEVCHAR2	Integer	Additional device characteristics.
DEVCLASS	Integer	Device class. (See Table DCLI–7 for a list of the values returned.)
DEVDEPEND	Integer	Device-dependent information.
DEVDEPEND2	Integer	Additional device-dependent information.
DEVLOCKNAM	String	A unique lock name for the device.
DEVNAM	String	Device name.
DEVSTS	Integer	Device-dependent status information.
DEVTYPE	Integer	Device type. (See Table DCLI–8 for a list of the values returned.)
DEVICE_TYPE_NAME	String	Device type name.

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
DFS_ACCESS	String	TRUE or FALSE to indicate whether the device is a virtual disk connected to a remote Distributed File System (DFS) server.
DIR	String	TRUE or FALSE to indicate whether the device is directory structured.
DMT	String	TRUE or FALSE to indicate whether the device is marked for dismount.
DUA	String	TRUE or FALSE to indicate whether the device is a generic device.
ELG	String	TRUE or FALSE to indicate whether the device has error logging enabled.
ERRCNT	Integer	Error count.
EXISTS	String	TRUE or FALSE to indicate whether the device exists on the system.
FOD	String	TRUE or FALSE to indicate whether the device is a files-oriented device.
FOR	String	TRUE or FALSE to indicate whether the device is mounted using the /FOREIGN qualifier.
FREEBLOCKS	Integer	Number of free blocks on the volume (disks only).
FULLDEVNAM	String	Fully qualified device name.
GEN	String	TRUE or FALSE to indicate whether the device is a generic device.
HOST_AVAIL	String	TRUE or FALSE to indicate whether the host serving the primary path is available.
HOST_COUNT	Integer	Number of hosts that make the device available to other nodes in the VMScluster.
HOST_NAME	String	Name of the host serving the primary path.
HOST_TYPE	String	Hardware type of the host serving the primary path.
IDV	String	TRUE or FALSE to indicate whether the device is capable of providing input.
LOCKID	Integer	Clusterwide lock identification.
LOGVOLNAM	String	Logical volume name.
MAXBLOCK	Integer	Number of logical blocks on the volume.
MAXFILES	Integer	Maximum number of files on the volume (disks only).
MBX	String	TRUE or FALSE to indicate whether the device is a mailbox.
MEDIA_ID	String	Nondecoded media ID.

[1] In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
MEDIA_NAME	String	Either the name of the disk or the tape type.
MEDIA_TYPE	String	Device name prefix.
MNT	String	TRUE or FALSE to indicate whether the device is mounted.
MOUNTCNT	Integer	Mount count.
NET	String	TRUE or FALSE to indicate whether the device is a network device.
NEXTDEVNAM	String	Device name of the next volume in a volume set (disks only).
ODV	String	TRUE or FALSE to indicate whether the device is capable of providing output.
OPCNT	Integer	Operation count.
OPR	String	TRUE or FALSE to indicate whether the device is an operator.
OWNUIC	String	User identification code (UIC) of the device owner.
PID	String	Process identification number of the device owner.
RCK	String	TRUE or FALSE to indicate whether the device has read checking enabled.
RCT	String	TRUE or FALSE to indicate whether the disk contains RCT.
REC	String	TRUE or FALSE to indicate whether the device is record oriented.
RECSIZ	Integer	Blocked record size.
REFCNT	Integer	Reference count of processes using the device.
REMOTE_DEVICE	String	TRUE or FALSE to indicate whether the device is a remote device.
RND	String	TRUE or FALSE to indicate whether the device allows random access.
ROOTDEVNAM	String	Device name of the root volume in a volume set (disks only).
RTM	String	TRUE or FALSE to indicate whether the device is a real-time device.
SDI	String	TRUE or FALSE to indicate whether the device is single-directory structured.
SECTORS	Integer	Number of sectors per track (disks only).
SERIALNUM	Integer	Volume serial number (disks only).
SERVED_DEVICE	String	TRUE or FALSE to indicate whether the device is a served device.

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
SET_HOST_TERMINAL	String	TRUE or FALSE to indicate whether the device is a remote terminal for a SET HOST session from a remote node.
SHDW_CATCHUP_ COPYING	String	TRUE or FALSE to indicate whether the device is a member that is the target of a full copy operation.
SHDW_MASTER	String	TRUE or FALSE to indicate whether the device is a virtual unit.
SHDW_MASTER_NAME	String	Device name of the virtual unit that represents the shadow set of which the specified device is a member. F$GETDVI returns a null string ("") if the specified device is not a member, or is itself a virtual unit.
SHDW_MEMBER	String	TRUE or FALSE to indicate whether the device is a shadow set member.
SHDW_MERGE_COPYING	String	TRUE or FALSE to indicate whether the device is a merge member of the shadow set.
SHDW_NEXT_MBR_NAME	String	Device name of the next member in the shadow set. If you specify a virtual unit or a member, F$GETDVI returns the device name of members in random order. If you specify the name of a device that is neither a virtual unit nor a member, F$GETDVI returns a null string. F$GETDVI returns the device name of the next member in the shadow set even if the next member has been removed from the shadow set. The device name includes the allocation class if the allocation class is not zero; otherwise it includes the device name of the disk controller.
SHR	String	TRUE or FALSE to indicate whether the device is shareable.
SPL	String	TRUE or FALSE to indicate whether the device is being spooled.
SPLDEVNAM	String	Name of the device being spooled.
SQD	String	TRUE or FALSE to indicate whether the device is sequential block-oriented (that is, magnetic tape).
STS	Integer	Status information.
SWL	String	TRUE or FALSE to indicate whether the device is software write-locked.
TRACKS	Integer	Number of tracks per cylinder (disks only).
TRANSCNT	Integer	Volume transaction count.

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
TRM	String	TRUE or FALSE to indicate whether the device is a terminal.
TT_ACCPORNAM	String	The terminal server name and port name.
TT_ALTYPEAHD	String	TRUE or FALSE to indicate whether the terminal has an alternate type-ahead buffer (terminals only).
TT_ANSICRT	String	TRUE or FALSE to indicate whether the terminal is an ANSI CRT terminal (terminals only).
TT_APP_KEYPAD	String	TRUE or FALSE to indicate whether the keypad is in applications mode (terminals only).
TT_AUTOBAUD	String	TRUE or FALSE to indicate whether the terminal has automatic baud rate detection (terminals only).
TT_AVO	String	TRUE or FALSE to indicate whether the terminal has a VT100-family terminal display (terminals only).
TT_BLOCK	String	TRUE or FALSE to indicate whether the terminal has block mode capability (terminals only).
TT_BRDCSTMBX	String	TRUE or FALSE to indicate whether the terminal uses mailbox broadcast messages (terminals only).
TT_CHARSET	Integer	A bitmap indicating the coded character set supported by the terminal.
TT_CRFILL	String	TRUE or FALSE to indicate whether the terminal requires fill after a carriage return (terminals only).
TT_CS_KANA	String	TRUE or FALSE to indicate whether the terminal supports the DEC Kana coded character set.
TT_CS_KANJI	String	TRUE or FALSE to indicate whether the terminal supports the DEC Kanji coded character set.
TT_CS_HANZI	String	TRUE or FALSE to indicate whether the terminal supports the DEC Hanzi coded character set.
TT_CS_HANGUL	String	TRUE or FALSE to indicate whether the terminal supports the DEC Korean coded character set.
TT_CS_HANYU	String	TRUE or FALSE to indicate whether the terminal supports the DEC Hanyu coded character set.
TT_CS_THAI	String	TRUE or FALSE to indicate whether the terminal supports the DEC Thai coded character set.
TT_DECCRT	String	TRUE or FALSE to indicate whether the terminal is a Digital CRT terminal (terminals only).
TT_DECCRT2	String	TRUE or FALSE to indicate whether the terminal is a Digital CRT2 terminal (terminals only).
TT_DECCRT3	String	TRUE or FALSE to indicate whether the terminal is a Digital CRT3 terminal (terminals only).

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
TT_DECCRT4	String	TRUE or FALSE to indicate whether the terminal is a Digital CRT4 terminal (terminals only).
TT_DIALUP	String	TRUE or FALSE to indicate whether the terminal is connected to dialup (terminals only).
TT_DISCONNECT	String	TRUE or FALSE to indicate whether the terminal can be disconnected (terminals only).
TT_DMA	String	TRUE or FALSE to indicate whether the terminal has direct memory access (DMA) mode (terminals only).
TT_DRCS	String	TRUE or FALSE to indicate whether the terminal supports loadable character fonts (terminals only).
TT_EDIT	String	TRUE or FALSE to indicate whether the edit characteristic is set.
TT_EDITING	String	TRUE or FALSE to indicate whether advanced editing is enabled (terminals only).
TT_EIGHTBIT	String	TRUE or FALSE to indicate whether the terminal uses the 8-bit ASCII character set (terminals only).
TT_ESCAPE	String	TRUE or FALSE to indicate whether the terminal generates escape sequences (terminals only).
TT_FALLBACK	String	TRUE or FALSE to indicate whether the terminal uses the multinational fallback option (terminals only).
TT_HALFDUP	String	TRUE or FALSE to indicate whether the terminal is in half-duplex mode (terminals only).
TT_HANGUP	String	TRUE or FALSE to indicate whether the hangup characteristic is set (terminals only).
TT_HOSTSYNC	String	TRUE or FALSE to indicate whether the terminal has host/terminal communication (terminals only).
TT_INSERT	String	TRUE or FALSE to indicate whether insert mode is the default line editing mode (terminals only).
TT_LFFILL	String	TRUE or FALSE to indicate whether the terminal requires fill after a line feed (terminals only).
TT_LOCALECHO	String	TRUE or FALSE to indicate whether the local echo characteristic is set (terminals only).
TT_LOWER	String	TRUE or FALSE to indicate whether the terminal has the lowercase characters set (terminals only).
TT_MBXDSABL	String	TRUE or FALSE to indicate whether mailboxes associated with the terminal will receive unsolicited input notification or input notification (terminals only).

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
TT_MECHFORM	String	TRUE or FALSE to indicate whether the terminal has mechanical form feed (terminals only).
TT_MECHTAB	String	TRUE or FALSE to indicate whether the terminal has mechanical tabs and is capable of tab expansion (terminals only).
TT_MODEM	String	TRUE or FALSE to indicate whether the terminal is connected to a modem (terminals only).
TT_MODHANGUP	String	TRUE or FALSE to indicate whether the modify hangup characteristic is set (terminals only).
TT_NOBRDCST	String	TRUE or FALSE to indicate whether the terminal will receive broadcast messages (terminals only).
TT_NOECHO	String	TRUE or FALSE to indicate whether the input characters are echoed.
TT_NOTYPEAHD	String	TRUE or FALSE to indicate whether data must be solicited by a read operation.
TT_OPER	String	TRUE or FALSE to indicate whether the terminal is an operator terminal (terminals only).
TT_PAGE	Integer	Terminal page length (terminals only).
TT_PASTHRU	String	TRUE or FALSE to indicate whether PASSALL mode with flow control is available (terminals only).
TT_PHYDEVNAM	String	Physical device name associated with a channel number or virtual terminal.
TT_PRINTER	String	TRUE or FALSE to indicate whether there is a printer port available (terminals only).
TT_READSYNC	String	TRUE or FALSE to indicate whether the terminal has read synchronization (terminals only).
TT_REGIS	String	TRUE or FALSE to indicate whether the terminal has ReGIS graphics (terminals only).
TT_REMOTE	String	TRUE or FALSE to indicate whether the terminal has established modem control (terminals only).
TT_SCOPE	String	TRUE or FALSE to indicate whether the terminal is a video screen display (terminals only).
TT_SECURE	String	TRUE or FALSE to indicate whether the terminal can recognize the secure server (terminals only).
TT_SETSPEED	String	TRUE or FALSE to indicate whether you cannot set the speed on the terminal line (terminals only).
TT_SIXEL	String	TRUE or FALSE to indicate whether the sixel is supported (terminals only).
TT_SYSPWD	String	TRUE or FALSE to indicate whether the system password is enabled for a particular terminal.

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

(continued on next page)

Table DCLI–6 (Cont.) F$GETDVI Items

Item	Return Type	Information Returned[1]
TT_TTSYNC	String	TRUE or FALSE to indicate whether there is terminal/host synchronization (terminals only).
TT_WRAP	String	TRUE or FALSE to indicate whether a new line should be inserted if the cursor moves beyond the right margin.
UNIT	Integer	The unit number.
VOLCOUNT	Integer	The count of volumes in a volume set (disks only).
VOLNAM	String	The volume name.
VOLNUMBER	Integer	Number of the current volume in a volume set (disks only).
VOLSETMEM	String	TRUE or FALSE to indicate whether the device is a volume set (disks only).
VPROT	String	The volume protection mask.
WCK	String	TRUE or FALSE to indicate whether the device has write checking enabled.

[1]In addition to the return information listed, the F$GETDVI lexical function returns any error messages generated by the system service $GETDVI.

Table DCLI–7 lists the values returned by the DEVCLASS item.

Table DCLI–7 Values Returned by the DEVCLASS Item

Value	Device Class	Symbolic Name
1	Disk device	DC$_DISK
2	Tape device	DC$_TAPE
32	Synchronous communications device	DC$_SCOM
65	Card reader	DC$_CARD
66	Terminal	DC$_TERM
67	Line printer	DC$_LP
70	Workstation	DC$_WORKSTATION
96	Real-time	DC$_REALTIME
97	DECvoice	DC$_DECVOICE
128	Bus	DC$_BUS
160	Mailbox	DC$_ MAILBOX
170	Remote console storage	DC$_REMCSL_STORAGE
200	Miscellaneous device	DC$_MISC

Table DCLI–8 lists the values returned by the DEVTYPE item.

Table DCLI–8 Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_DISK			
1	RK06	54	GENERIC_DK
2	RK07	55	RX23
3	RP04	56	RF31
4	RP05	57	RF72
5	RP06	58	RAM_DISK
6	RM03	59	RZ25
7	RP07	60	RZ56
8	RP07HT	61	RZ57
9	RL01	62	RX23S
10	RL02	63	RX33S
11	RX02	64	RA92
12	RX04	65	SSTRIPE
13	RM80	66	RZ23L
14	TU58	67	RX26
15	RM05	68	RZ57I
16	RX01	69	RZ31
17	ML11	70	RZ58
18	RB02	71	SCSI-MO
19	RB80	71	RWZ01
20	RA80	72	LO_COST_CD
21	RA81	72	RRD42
22	RA60	73	CD_LOADER_1
23	RC25	74	ESE25
23	RZ01	75	RFH31
24	RZF01	76	RFH72
24	RCF25	77	RF73
25	RD51	78	RFH73
26	RX50	79	RA72
27	RD52	80	RA71
28	RD53	80	RAH72
29	RD26	81	RF32
30	RA82	82	RFH32
31	RD31	83	RFF31
32	RD54	83	RF31F
33	CRX50	84	RZ72
34	RRD50	85	RZ73
35	GENERIC_DU	86	RZ35

(continued on next page)

Table DCLI–8 (Cont.) Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_DISK			
36	RX33	87	RZ24L
37	RX18	88	RZ25L
38	RA70	89	RZ55L
39	RA90	90	RZ56L
40	RD32	91	RZ57L
41	DISK9	92	RA73
42	RX35	93	RZ26
43	RF30	94	RZ36
44	RF70	95	RZ74
44	RF71	96	ESE52
45	RD33	97	ESE56
46	ESE20	98	ESE58
47	TU56	129	FD1
48	RZ22	130	FD2
49	RZ23	131	FD3
50	RZ24	132	FD4
51	RZ55	133	FD5
52	RRD40	134	FD6
53	RRD40		

(continued on next page)

Table DCLI–8 (Cont.) Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_TAPE			
1	TE16	24	RV60
2	TU45	25	TZ30
3	TU77	26	TM32
4	TS11	27	TZX0
5	TU78	27	TSZ05
6	TA78	28	GENERIC_MK
7	TU80	29	TK50S
8	TU81	30	TZ30S
9	TA81	31	TK70L
10	TK50	32	TLZ04
11	MR_TU70	33	TZK10
12	MR_TU72	34	TSZ07
13	MW_TSU05	35	TSZ08
14	MW_TSV05	36	TA90E
15	TK70	37	TZK11
16	RV20	38	TZ85
16	RV80	39	TZ86
17	TK60	40	TZ87
18	GENERIC_TU	41	TZ857
19	TA79	42	EXABYTE
20	TAPE9	43	TAPE_LOADER_1
21	TA90	44	TA91
22	TF30	45	TLZ06
23	TF70	46	TA85

(continued on next page)

Table DCLI–8 (Cont.) Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_SCOM			
1	DMC11	27	DMB32
2	DMR11	28	YI_KMS11K
3	XK_3271	29	ET_DEBNT
4	XJ_2780	29	ET_DEBNA
5	NW_X25	30	SJ_DSV11
6	NV_X29	31	SL_DSB32
7	SB_ISB11	32	ZS_DST32
8	MX_MUX200	33	XQ_DELQA
9	DMP11	34	ET_DEBNI
10	DMF32	35	EZ_SGEC
11	XV_3271	36	EX_DEMNA
12	CI	37	DIV32
13	NI	38	ZQ_DEQTA
14	UNA11	39	FT_NI
14	DEUNA	40	EP_LANCE
15	YN_X25	41	KWV32
16	YO_X25	42	SM_DSF32
17	YP_ADCCP	43	FX_DEMFA
18	YQ_3271	44	SF_DSF32
19	YR_DDCMP	45	SE_DUP11
20	YS_SDLC	46	SE_DPV11
21	UK_KTC32	47	ZT_DSW
22	DEQNA	48	FC_DEFZA
23	DMV11	49	EC_PMAD
24	ES_LANCE	50	EZ_TGEC
25	DELUA	51	EA_DEANA
26	NQ_3271		
Device Class: DC$_CARD			
1	CR11		

(continued on next page)

Table DCLI–8 (Cont.) Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_TERM			
0	TTYUNKN	42	LN01K
1	VT05	43	LA80
2	VK100	64	VT5X
3	VT173	64	VT52
4	TQ_BTS	65	VT55
10	TEK401X	66	DZ11
16	FT1	67	DZ32
17	FT2	68	DZ730
18	FT3	69	DMZ32
19	FT4	70	DHV
20	FT5	71	DHU
21	FT6	72	SLU
22	FT7	73	TERM9
23	FT8	74	LAT
32	LAX	96	VT100
33	LA36	97	VT101
33	LA120	98	VT102
34	LA34	99	VT105
35	LA38	100	VT125
36	LA12	101	VT131
37	LA24	102	VT132
37	LA100	110	VT200_Series
38	LQP02	111	Pro_Series
40	LA210	112	VT300_Series
41	LN03	113	VT400_Series
Device Class: DC$_LP			
1	LP11	5	LI_DMB32
2	LA11	6	PRTR9
3	LA180	7	SCSI_SCANNER_1
4	LC_DMF32		
Device Class: DC$_WORKSTATION			
1	VAXstation 100	4	VAXstation virtual device
2	VAXstation 125	5	DECW output device
3	VAXstation 8200	6	DECW input device

(continued on next page)

Table DCLI–8 (Cont.) Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_REALTIME			
1	LPA11	9	XP_PCL11B
2	DR780	10	IX_IEX11
3	DR750	11	FP_FEPCM
4	DR11W	12	TK_FCM
5	PCL11R	13	XI_DR11C
6	PCL11T	14	XA_DRV11WA
7	DR11C	15	DRB32
8	XI_DR11C	16	HX_DRQ3B
Device Class: DC$_DECVOICE			
1	VQ class driver	3	VM M3135 port driver
2	VX M7132 port driver	4	VN M3136 port driver
Device Class: DC$_BUS			
1	CI780	23	RV20P
2	CI750	23	RV80P
3	UQPORT	24	TK60P
3	UDA50	25	SII
4	UDA50A	26	KFSQSA
5	LESI	26	KFQSA
6	TU81P	27	SHAC
7	RDRX	28	CIXCD
8	TK50P	29	N5380
9	RUX50P	30	SCSII
10	RC26P	31	HSX50
11	QDA50	31	KDM70
11	KDA50	32	TM32P
12	BDA50	33	TK7LP
12	KDB50	34	SWIFT
13	RRD50P	35	N53C94
14	QDA25	36	KFMSA
14	KDA25	37	SCSI_XTENDR
15	BC1750	38	FT_TRACE_RAM
16	BCA	39	XVIB
17	RQDX3	40	XZA_SCSI

(continued on next page)

Table DCLI–8 (Cont.) Values Returned by the DEVTYPE Item

Value	Device Type	Value	Device Type
Device Class: DC$_BUS			
18	NISCA	41	XZA_DSSI
19	AIO	42	N710_SCSI
19	KFBTA	43	N710_DSSI
20	AIE	44	AHA1742A
20	DEBNT	45	TZA_SCSI
21	BSA	46	N810_SCSI
21	KSB50	47	CIPCA
22	TK709	48	ISP1020
Device Class: DC$_MAILBOX			
1	MBX	3	NULL
2	SHRMBX	4	PIPE
Device Class: DC$_REMCSL_STORAGE			
DAP-accessed device	1		
Device Class: DC$_MISC			
1	DN11	3	SFUN9
2	PV	4	USER9

Example

```
$ ERR = F$GETDVI("_DQA0","ERRCNT")
$ SHOW SYMBOL ERR
  ERR = 0  Hex = 00000000 Octal = 000000
```

This example shows how to use the F$GETDVI function to return an error count for the device DQA0. You must place quotation marks (‘‘ ’’) around the device name DQA0 and the item ERRCNT because they are string literals.

F$GETJPI

Returns information about the specified process.

Requires GROUP privilege to obtain information on other processes in the same group. Requires WORLD privilege to obtain information on any other processes in the system.

Format

F$GETJPI(pid,item)

Return Value

Either an integer or a character string, depending on the item you request. Table DCLI–9 shows the data types of the values returned for each item.

Arguments

pid

Specifies the process identification (PID) number of the process for which information is being reported. Specify the **pid** argument as a character string expression. You can omit the leading zeros.

If you specify a null string (""), the current PID number is used.

You cannot use an asterisk (*) or percent sign (%) wildcard character to specify the **pid** argument in the F$GETJPI function, as you can with the $GETJPI system service. To get a list of process identification numbers, use the F$PID function.

item

Indicates the type of process information to be returned. Specify the **item** argument as a character string expression. You can specify any one of the items listed in Table DCLI–9.

Description

The F$GETJPI lexical function invokes the $GETJPI system service to return information about the specified process. The function returns information on all items that can be specified with the $GETJPI system service. For more information on the $GETJPI system service, see the *OpenVMS System Services Reference Manual*.

You can use the F$GETJPI lexical function to find out whether a process automatically unshelves files.

When you specify the STS2 item code, F$GETJPI returns a 32–bit numeric value. When you convert this numeric value to binary format, the digit at symbolic bit position PCB$V_NOUNSHELVE shows you the process unshelving default. If the bit is 1, automatic unshelving is turned off; if 0, automatic unshelving is turned on.

Table DCLI–9 lists the items you can specify with the F$GETJPI function, the information returned, and the data type of this information.

Table DCLI–9 F$GETJPI Items

Item	Return Type	Information Returned
ACCOUNT	String	Account name string (8 characters filled with trailing blanks).
APTCNT	Integer	Active page table count.
ASTACT	Integer	Access modes with active asynchronous system traps (ASTs).
ASTCNT	Integer	Remaining AST quota.
ASTEN	Integer	Access modes with ASTs enabled.
ASTLM	Integer	AST limit quota.
AUTHPRI	Integer	Maximum priority that a process without the ALTPRI (alter priority) privilege can achieve with the $SETPRI system service.
AUTHPRIV	String	Privileges that a process is authorized to enable.
BIOCNT	Integer	Remaining buffered I/O quota.
BIOLM	Integer	Buffered I/O limit quota.
BUFIO	Integer	Count of process buffered I/O operations.
BYTCNT	Integer	Remaining buffered I/O byte count quota.
BYTLM	Integer	Buffered I/O byte count quota.
CLINAME	String	Current command language interpreter; always returns DCL.
CPULIM	Integer	Limit on process CPU time.
CPUTIM	Integer	CPU time used in hundredths of a second.
CREPRC_FLAGS	Integer	Flags specified by the **stsflg** argument in the $CREPRC call that created the process.
CURPRIV	String	Current process privileges.
DFPFC	Integer	Default page fault cluster size.
DFWSCNT	Integer	Default working set size.
DIOCNT	Integer	Remaining direct I/O quota.
DIOLM	Integer	Direct I/O limit quota.
DIRIO	Integer	Count of direct I/O operations for the process.
EFCS	Integer	Local event flags 0–31.
EFCU	Integer	Local event flags 32–63.
EFWM	Integer	Event flag wait mask.
ENQCNT	Integer	Lock request quota remaining.
ENQLM	Integer	Lock request quota limit.

(continued on next page)

Table DCLI–9 (Cont.) F$GETJPI Items

Item	Return Type	Information Returned
EXCVEC	Integer	Address of a list of exception vectors.
FAST_VP_SWITCH	Integer	Number of times this process has issued a vector instruction that enabled an inactive vector processor without the expense of a vector context switch.
FILCNT	Integer	Remaining open file quota.
FILLM	Integer	Open file quota.
FINALEXC	Integer	Address of a list of final exception vectors.
FREP0VA	Integer	First free page at end of program region (P0 space) (irrelevant if no image is running).
FREP1VA	Integer	First free page at end of control region (P1 space).
FREPTECNT	Integer	Number of pages available for virtual memory expansion.
GPGCNT	Integer	Global page count in working set.
GRP	Integer	Group number of the user identification code (UIC).
IMAGECOUNT	Integer	Number of images that have been run down for the process.
IMAGNAME	String	File name of the current image.
IMAGPRIV	String	Privileges with which the current image was installed.
JOBPRCCNT	Integer	Number of subprocesses owned by the job.
JOBTYPE	Integer	Execution mode of the process at the root of the job tree.
LAST_LOGIN_I	String	Time of your last interactive login (the value that was reported when you logged in).
LAST_LOGIN_N	String	Time of your last noninteractive login (the value that was reported when you logged in).
LOGIN_FAILURES	Integer	Number of login failures that occurred prior to the start of the current session (the value that was reported when you logged in).
LOGIN_FLAGS	Integer	A longword bitmask that contains additional information relating to the login sequence.
LOGINTIM	String	Process creation time.

(continued on next page)

Table DCLI–9 (Cont.) F$GETJPI Items

Item	Return Type	Information Returned
MASTER_PID	String	Process identification (PID) number of the process at the top of the current job's process tree.
MAXDETACH	Integer	Maximum number of detached processes allowed the user who owns the process.
MAXJOBS	Integer	Maximum number of active processes allowed for the user who owns the process.
MEM	Integer	Member number of the UIC.
MODE	String	Current process mode (BATCH, INTERACTIVE, NETWORK, or OTHER).
MSGMASK	Integer	Default message mask.
NODENAME	String	The name of the VMScluster node on which the process is running.
NODE_CSID	Integer	Cluster ID of the VMScluster node on which the process is running.
NODE_VERSION	String	Operating system version number of the VMScluster node on which the process is running.
OWNER	String	Process identification number of process owner.
PAGEFLTS	Integer	Count of page faults.
PAGFILCNT	Integer	Remaining paging file quota.
PAGFILLOC	Integer	Location of the paging file.
PGFLQUOTA	Integer	Paging file quota (maximum virtual page count).
PHDFLAGS	Integer	Flags word.
PID	String	Process identification number.
PPGCNT	Integer	Process page count.
PRCCNT	Integer	Number of subprocesses owned by the process.
PRCLM	Integer	Subprocess quota.
PRCNAM	String	Process name.
PRI	Integer	Process's current priority.
PRIB	Integer	Process's base priority.
PROC_INDEX	Integer	Process's index number.
PROCESS_RIGHTS	String	The contents of the process's local rights list, including your UIC. This item code returns a list of identifier names separated by commas (,).

(continued on next page)

Table DCLI–9 (Cont.) F$GETJPI Items

Item	Return Type	Information Returned
PROCPRIV	String	Process's default privileges.
RIGHTSLIST	String	The contents of all of the process rights lists; the equivalent of PROCESS_ RIGHTS plus SYSTEM_RIGHTS. This item code returns a list of identifier names separated by commas.
RIGHTS_SIZE	Integer	Number of bytes required to buffer the rights list. The rights list includes both the system rights list and the process rights list.
SHRFILLM	Integer	Maximum number of open shared files allowed for the job to which the process belongs.
SITESPEC	Integer	Per-process site-specific longword.
SLOW_VP_SWITCH	Integer	Number of times this process has issued a vector instruction that enabled an inactive vector processor with a full vector context switch.
STATE	String	Process state.
STS	Integer	First longword of process status flags.
STS2	Integer	Second longword of process status flags.
SWPFILLOC	Integer	Location of the swap file.
SYSTEM_RIGHTS	String	The contents of the system rights list for the process. This item code returns a list of identifier names separated by commas.
TABLENAME	String	File specification of the process's current command language interpreter (CLI) table.
TERMINAL	String	Login terminal name for interactive users (1–7 characters).
TMBU	Integer	Termination mailbox unit number.
TQCNT	Integer	Remaining timer queue entry quota.
TQLM	Integer	Timer queue entry quota.
TT_ACCPORNAM	String	Access port name for the terminal associated with the process.
TT_PHYDEVNAM	String	Physical device name of the terminal associated with the process.
UAF_FLAGS	Integer	User authorization file (UAF) flags from the UAF record of the user who owns the process.
UIC	String	Process's user identification code (UIC).

(continued on next page)

Table DCLI–9 (Cont.) F$GETJPI Items

Item	Return Type	Information Returned
USERNAME	String	User name string (12 characters filled with trailing blanks).
VIRTPEAK	Integer	Peak virtual address size.
VOLUMES	Integer	Count of currently mounted volumes.
VP_CONSUMER	Boolean	Flag indicating whether the process is a vector consumer.
VP_CPUTIM	Integer	Total amount of time the process has accumulated as a vector customer.
WSAUTH	Integer	Maximum authorized working set size.
WSAUTHEXT	Integer	Maximum authorized working set extent.
WSEXTENT	Integer	Current working set extent.
WSPEAK	Integer	Working set peak.
WSQUOTA	Integer	Working set size quota.
WSSIZE	Integer	Process's current working set size.

If you use the F$GETJPI function to request information on the null process or the swapper process, you can specify any of the items in Table DCLI–9 except the following:

ACCOUNT	BYTLM	ENQCNT	ENQLM
EXCVEC	FILCNT	FILM	FINALEXC
IMAGNAME	LOGINTIM	MSGMASK	PAGFILCNT
PGFLQUOTA	PRCCNT	PRCLM	PROCPRIV
SITESPEC	TQCNT	TQLM	USERNAME
VIRTPEAK	VOLUMES	WSPEAK	

Example

```
$ NAME = F$GETJPI("3B0018","USERNAME")
$ SHOW SYMBOL NAME
  NAME = "JANE"
```

This example shows how to use the F$GETJPI function to return the user name for the process number 3B0018. The user name is assigned to the symbol NAME.

F$GETQUI

Returns information about queues, including batch and print jobs currently in the queues, form definitions, and characteristic definitions kept in the queue database.

Also returns information about queue managers.

For most operations, read (R) access is required.

Format

F$GETQUI(function,[item],[object-id],[flags])

Return Value

Either an integer or a character string, depending on the item you request. For items that return a Boolean value, the string is TRUE or FALSE. If the $GETQUI system service returns an error code, F$GETQUI returns a null string ("").

Arguments

function

Specifies the action that the F$GETQUI lexical function is to perform. F$GETQUI supports all functions that can be specified with the $GETQUI system service. The following table lists these functions:

Function	Description
CANCEL_OPERATION	Terminates any wildcard operation that may have been initiated by a previous call to F$GETQUI.
DISPLAY_CHARACTERISTIC	Returns information about a specific characteristic definition or the next characteristic definition in a wildcard operation.
DISPLAY_ENTRY	Returns information about a specific job entry or the next job entry that matches the selection criteria in a wildcard operation. The DISPLAY_ENTRY function code is similar to the DISPLAY_JOB function code in that both return job information. DISPLAY_JOB, however, requires that a call be made to establish queue context; DISPLAY_ENTRY does not require that queue context be established. Only those entries that match the user-name of the current process will be processed.
DISPLAY_FILE	Returns information about the next file defined for the current job context. Before you make a call to F$GETQUI to request file information, you must make a call to display queue and job information (with the DISPLAY_QUEUE and DISPLAY_JOB function codes) or to display entry information (with the DISPLAY_ENTRY function code).
DISPLAY_FORM	Returns information about a specific form definition or the next form definition in a wildcard operation.

Function	Description
DISPLAY_JOB	Returns information about the next job defined for the current queue context. Before you make a call to F$GETQUI to request job information, you must make a call to display queue information (with the DISPLAY_QUEUE function code). The DISPLAY_JOB function code is similar to the DISPLAY_ENTRY function code in that both return job information. DISPLAY_JOB, however, requires that a call be made to establish queue context; DISPLAY_ENTRY does not require that queue context be established.
DISPLAY_MANAGER	Returns information about a specific queue manager or the next queue manager in a wildcard operation.
DISPLAY_QUEUE	Returns information about a specific queue definition or the next queue definition in a wildcard operation.
TRANSLATE_QUEUE	Translates a logical name for a queue to the equivalence name for the queue.

Some function arguments cannot be specified with the **item-code,** the **object-id,** or the **flags** argument. The following table lists each function argument and corresponding format line to show whether the **item-code**, **object-id** and **flags** arguments are required, optional, or not applicable for that specific function. In the following format lines, brackets ([ ]) denote an optional argument. An omitted argument means the argument is not applicable for that function. Note that two commas (,,) must be used as placeholders to denote an omitted (whether optional or not applicable) argument.

Function	Format Line
CANCEL_OPERATION	F$GETQUI(‘‘CANCEL_OPERATION’’) or F$GETQUI(‘‘ ’’)
DISPLAY_CHARACTERISTIC	F$GETQUI(‘‘DISPLAY_CHARACTERISTIC’’, [item],object-id,[flags])
DISPLAY_ENTRY	F$GETQUI(‘‘DISPLAY_ENTRY’’,[item], [object-id],[flags])
DISPLAY_FILE	F$GETQUI(‘‘DISPLAY_FILE’’,[item],,[flags])
DISPLAY_FORM	F$GETQUI(‘‘DISPLAY_FORM’’,[item], object-id,[flags])
DISPLAY_JOB	F$GETQUI(‘‘DISPLAY_JOB’’,[item],,[flags])
DISPLAY_MANAGER	F$GETQUI("DISPLAY_MANAGER",[item], object-id,[flags])
DISPLAY_QUEUE	F$GETQUI(‘‘DISPLAY_QUEUE’’,[item], object-id,[flags])
TRANSLATE_QUEUE	F$GETQUI(‘‘TRANSLATE_QUEUE’’,[item],object-id)

item

Corresponds to a $GETQUI system service output item code. The **item** argument specifies the kind of information you want returned about a particular queue, job, file, form, or characteristic. On VAX, queue manager information is also available. Table DCLI–11 lists each item code and the data type of the value returned for each item code.

object-id

Corresponds to the $GETQUI system service QUI$SEARCH_NAME, QUI$_SEARCH_NUMBER, and QUI$_SEARCH_JOB_NAME input item codes. The **object-id** argument specifies either the name or the number of an object (for example, a specific queue name, job name, or form number) about which

F$GETQUI is to return information. The asterisk (*) and the percent sign (%) wildcard characters are allowed for the following functions:

DISPLAY_CHARACTERISTIC
DISPLAY_ENTRY
DISPLAY_FORM
DISPLAY_MANAGER
DISPLAY_QUEUE

By specifying an asterisk (*) or percent sign (%) wildcard character as the **object-id** argument on successive calls, you can get status information about one or more jobs in a specific queue or about files within jobs in a specific queue. When a name is used with wildcard characters, each call returns information for the next object (queue, form, and so on) in the list. A null string ("") is returned when the end of the list is reached. A wildcard can represent only object names, not object numbers.

flags

Specifies a list of keywords, separated by commas, that corresponds to the flags defined for the $GETQUI system service QUI$_SEARCH_FLAGS input item code. (These flags are used to define the scope of the object search specified in the call to the $GETQUI system service.) Note that keywords in Table DCLI–10 can be used only with certain function codes.

Table DCLI–10 F$GETQUI Keywords

Keyword	Valid Function Code	Description
ALL_JOBS	DISPLAY_JOB	Requests that F$GETQUI search all jobs included in the established queue context. If you do not specify this flag, F$GETQUI returns information only about jobs that have the same user name as the caller.
BATCH	DISPLAY_QUEUE DISPLAY_ENTRY	Selects batch queues.
EXECUTING_JOBS	DISPLAY_ENTRY DISPLAY_ JOB	Selects executing jobs.
FREEZE_CONTEXT	DISPLAY_ CHARACTERISTIC DISPLAY_ENTRY DISPLAY_FILE DISPLAY_FORM DISPLAY_JOB DISPLAY_MANAGER DISPLAY_QUEUE	When in wildcard mode, prevents advance of wildcard context to the next object. If you do not specify this flag, the context is advanced to the next object.
GENERIC	DISPLAY_ENTRY DISPLAY_ QUEUE	Selects generic queues for searching.
HOLDING_JOBS	DISPLAY_ENTRY DISPLAY_ JOB	Selects jobs on unconditional hold.

(continued on next page)

Table DCLI–10 (Cont.) F$GETQUI Keywords

Keyword	Valid Function Code	Description
PENDING_JOBS	DISPLAY_ENTRY DISPLAY_ JOB	Selects pending jobs.
PRINTER	DISPLAY_QUEUE DISPLAY_ENTRY	Selects printer queues.
RETAINED_JOBS	DISPLAY_ENTRY DISPLAY_ JOB	Selects jobs being retained.
SERVER	DISPLAY_QUEUE DISPLAY_ENTRY	Selects server queues.
SYMBIONT	DISPLAY_QUEUE DISPLAY_ENTRY	Selects all output queues. Equivalent to specifying ''PRINTER,SERVER,TERMINAL''.
TERMINAL	DISPLAY_QUEUE DISPLAY_ENTRY	Selects terminal queues.
THIS_JOB	DISPLAY_FILE DISPLAY_ JOB DISPLAY_QUEUE	Selects all job file information about the calling batch job, the command file being executed, or the queue associated with the calling batch job.
TIMED_RELEASE_JOBS	DISPLAY_ENTRY DISPLAY_ JOB	Selects jobs on hold until a specified time.
WILDCARD	DISPLAY_ CHARACTERISTIC DISPLAY_ENTRY DISPLAY_FORM DISPLAY_MANAGER DISPLAY_QUEUE	Establishes and saves a context. Because the context is saved, the next operation can be performed based on that context.

Description

The F$GETQUI lexical function invokes the $GETQUI system service to return information about queues, batch and print jobs currently in those queues, form definitions, and characteristic definitions kept in the system job queue file. The F$GETQUI lexical function provides all the features of the $GETQUI system service, including wildcard and nested wildcard operations. For example, in nested wildcard operations, $GETQUI returns information about objects defined within another object. Specifically, this mode allows you to query jobs contained in a selected queue or files contained in a selected job in a sequence of calls. After each call, the system saves the GQC (internal GETQUI context block) so that the GQC can provide the queue or job context necessary for subsequent calls. For more information, see the description of the $GETQUI system service in the *OpenVMS System Services Reference Manual*.

The F$GETQUI function returns information on all items that can be specified with the $GETQUI system service. Table DCLI–11 lists the items you can specify with the F$GETQUI function, the information returned, and the data type of this information.

Table DCLI–11 F$GETQUI Items

Item	Return Type	Information Returned
ACCOUNT_NAME[1]	String	The account name of the owner of the specified job.
AFTER_TIME	String	The system time at or after which the specified job can execute.
ASSIGNED_QUEUE_NAME[1]	String	The name of the execution queue to which the logical queue specified in the call to F$GETQUI is assigned.
AUTOSTART_ON	String	A list of nodes or node device pairs indicating where the queue can start.
BASE_PRIORITY	Integer	The priority at which batch jobs are initiated from a batch execution queue or the priority of a symbiont process that controls output execution queues.
CHARACTERISTICS[1]	String	The characteristics associated with the specified queue or job.
CHARACTERISTIC_NAME	String	The name of the specified characteristic.
CHARACTERISTIC_NUMBER	Integer	The number of the specified characteristic.
CHECKPOINT_DATA[1]	String	The value of the DCL symbol BATCH$RESTART when the specified batch job is restarted.
CLI[1]	String	The name of the command language interpreter (CLI) used to execute the specified batch job. The file specification returned assumes the device name SYS$SYSTEM and the file type EXE.
COMPLETED_BLOCKS	Integer	The number of blocks that the symbiont has processed for the specified print job. This item code is applicable only to print jobs.
CONDITION_VECTOR[1]	Integer	The completion status of the specified job.
CPU_DEFAULT	String	The default CPU time limit specified for the queue in delta time. This item code is applicable only to batch execution queues.
CPU_LIMIT[1]	String	The maximum CPU time limit specified for the specified job or queue in delta time. This item code is applicable only to batch jobs and batch execution queues.
DEFAULT_FORM_NAME	String	The name of the default form associated with the specified output queue.
DEFAULT_FORM_STOCK	String	The name of the paper stock on which the specified default form is to be printed.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
DEVICE_NAME	String	The node and device (or both) on which the specified execution queue is located. For output execution queues, only the device name is returned. The node name is used only in dual-architecture VMScluster systems. The node name is specified by the system parameter SCSNODE for the processor on which the queue executes. For batch execution queues, a null string ("") is returned. To get the name of the node on which a batch queue is executing, use the SCSNODE_NAME item.
ENTRY_NUMBER	Integer	The queue entry number of the specified job.
EXECUTING_JOB_COUNT	Integer	The number of jobs in the queue that are currently executing.
FILE_BURST	String	TRUE or FALSE to indicate whether burst and flag pages are to be printed preceding a file.
FILE_CHECKPOINTED[1]	String	TRUE or FALSE to indicate whether the specified file is checkpointed.
FILE_COPIES[1]	Integer	The number of times the specified file is to be processed. This item code is applicable only to output execution queues.
FILE_COPIES_DONE[1]	Integer	The number of times the specified file has been processed. This item code is applicable only to output execution queues.
FILE_COUNT	Integer	The number of files in a specified job.
FILE_DELETE	String	TRUE or FALSE to indicate whether the specified file is to be deleted after execution of request.
FILE_DOUBLE_SPACE	String	TRUE or FALSE to indicate whether the symbiont formats the file with double spacing.
FILE_EXECUTING[1]	String	TRUE or FALSE to indicate whether the specified file is being processed.
FILE_FLAG	String	TRUE or FALSE to indicate whether a flag page is to be printed preceding a file.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
FILE_FLAGS[1]	Integer	The processing options that have been selected for the specified file. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of FILE_FLAGS: FILE_BURST FILE_DELETE FILE_DOUBLE_SPACE FILE_FLAG FILE_PAGE_HEADER FILE_PAGINATE FILE_PASSALL FILE_TRAILER
FILE_IDENTIFICATION[1]	String	The internal file-identification value that uniquely identifies the selected file. This value specifies (in order) the following three file-identification fields in the RMS NAM block: NAM$T_DVI (16 bytes) NAM$W_FID (6 bytes) NAM$W_DID (6 bytes)
FILE_PAGE_HEADER	String	TRUE or FALSE to indicate whether a page header is to be printed on each page of output.
FILE_PAGINATE	String	TRUE or FALSE to indicate whether the symbiont paginates output by inserting a form feed whenever output reaches the bottom margin of the form.
FILE_PASSALL	String	TRUE or FALSE to indicate whether the symbiont prints the file in PASSALL mode.
FILE_SETUP_MODULES[1]	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before the specified file is printed. This item code is meaningful only for output execution queues.
FILE_SPECIFICATION[1]	String	The fully qualified RMS file specification of the file about which F$GETQUI is returning information.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
FILE_STATUS[1]	Integer	File status information. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of FILE_STATUS: FILE_CHECKPOINTED FILE_EXECUTING
FILE_TRAILER	String	TRUE or FALSE to indicate whether a trailer page is to be printed following a file.
FIRST_PAGE[1]	Integer	The page number at which the printing of the specified file is to begin. This item code is applicable only to output execution queues.
FORM_DESCRIPTION	String	The text string that describes the specified form to users and operators.
FORM_FLAGS	Integer	The processing options that have been selected for the specified form. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of FORM_FLAGS: FORM_SHEET_FEED FORM_TRUNCATE FORM_WRAP
FORM_LENGTH	Integer	The physical length of the specified form in lines. This item code is applicable only to output execution queues.
FORM_MARGIN_BOTTOM	Integer	The bottom margin of the specified form in lines.
FORM_MARGIN_LEFT	Integer	The left margin of the specified form in characters.
FORM_MARGIN_RIGHT	Integer	The right margin of the specified form in characters.
FORM_MARGIN_TOP	Integer	The top margin of the specified form in lines.
FORM_NAME[1]	String	The name of the specified form or the mounted form associated with the specified job or queue.
FORM_NUMBER	Integer	The number of the specified form.
FORM_SETUP_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before a file is printed on the specified form. This item code is meaningful only for output execution queues.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
FORM_SHEET_FEED	String	TRUE or FALSE to indicate whether the symbiont pauses at the end of each physical page so that another sheet of paper can be inserted.
FORM_STOCK[1]	String	The name of the paper stock on which the specified form is to be printed.
FORM_TRUNCATE	String	TRUE or FALSE to indicate whether the printer discards any characters that exceed the specified right margin.
FORM_WIDTH	Integer	The width of the specified form.
FORM_WRAP	String	TRUE or FALSE to indicate whether the printer prints any characters that exceed the specified right margin on the following line.
GENERIC_TARGET	String	The names of the execution queues that are enabled to accept work from the specified generic queue. This item code is meaningful only for generic queues.
HOLDING_JOB_COUNT	Integer	The number of jobs in the queue being held until explicitly released.
INTERVENING_BLOCKS	Integer	The number of blocks associated with pending jobs in the queue that were skipped during the current call to F$GETQUI. These jobs were not reported because they did not match the selection criterion in effect for the call to F$GETQUI.
INTERVENING_JOBS	Integer	The number of of pending jobs in the queue that were skipped during the current call to F$GETQUI. These jobs were not reported because they did not match the selection criterion in effect for the call to F$GETQUI.
JOB_ABORTING	String	TRUE or FALSE to indicate whether the system is attempting to abort the execution of a job.
JOB_COMPLETION_QUEUE[1]	String	The name of the queue on which the specified job executed.
JOB_COMPLETION_TIME[1]	String	The time at which the execution of the specified job completed.
JOB_COPIES[1]	Integer	The number of times the specified print job is to be repeated.
JOB_COPIES_DONE[1]	Integer	The number of times that the specified print job has been repeated.
JOB_CPU_LIMIT[1]	String	TRUE or FALSE to indicate whether a CPU time limit is specified for the job.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
JOB_ERROR_RETENTION[1]	String	TRUE or FALSE to indicate whether the user requested that the specified job be retained in the queue if the job completes unsuccessfully.
JOB_EXECUTING	String	TRUE or FALSE to indicate whether the specified job is executing or printing.
JOB_FILE_BURST[1]	String	TRUE or FALSE to indicate whether a burst page option is explicitly specified for the job.
JOB_FILE_BURST_ONE[1]	String	TRUE or FALSE to indicate whether burst and flag pages precede only the first copy of the first file in the job.
JOB_FILE_FLAG[1]	String	TRUE or FALSE to indicate whether a flag page precedes each file in the job.
JOB_FILE_FLAG_ONE[1]	String	TRUE or FALSE to indicate whether a flag page precedes only the first copy of the first file in the job.
JOB_FILE_PAGINATE[1]	String	TRUE or FALSE to indicate whether a paginate option is explicitly specified for the job.
JOB_FILE_TRAILER[1]	String	TRUE or FALSE to indicate whether a trailer page follows each file in the job.
JOB_FILE_TRAILER_ONE[1]	String	TRUE or FALSE to indicate whether a trailer page follows only the last copy of the last file in the job.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
JOB_FLAGS[1]	Integer	The processing options selected for the specified job. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of JOB_FLAGS: JOB_CPU_LIMIT JOB_ERROR_RETENTION JOB_FILE_BURST JOB_FILE_BURST_ONE JOB_FILE_FLAG JOB_FILE_FLAG_ONE JOB_FILE_PAGINATE JOB_FILE_TRAILER JOB_FILE_TRAILER_ONE JOB_LOG_DELETE JOB_LOG_NULL JOB_LOG_NULL JOB_LOG_SPOOL JOB_LOWERCASE JOB_NOTIFY JOB_RESTART JOB_RETENTION_TIME JOB_WSDEFAULT JOB_WSEXTENT JOB_WSQUOTA
JOB_HOLDING	String	TRUE or FALSE to indicate whether the job will be held until it is explicitly released.
JOB_INACCESSIBLE	String	TRUE or FALSE to indicate whether the caller does not have read access to the specific job and file information in the system queue file. When FALSE, the DISPLAY_JOB and DISPLAY_FILE operations can return information for only the following output value item codes: AFTER_TIME COMPLETED_BLOCKS ENTRY_NUMBER INTERVENING_BLOCKS INTERVENING_JOBS JOB_SIZE JOB_STATUS
JOB_LIMIT	Integer	The number of jobs that can execute simultaneously on the specified queue. This item code is applicable only to batch execution queues.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
JOB_LOG_DELETE[1]	String	TRUE or FALSE to indicate whether the log file is deleted after it is printed.
JOB_LOG_NULL[1]	String	TRUE or FALSE to indicate whether a log file is not created.
JOB_LOG_SPOOL[1]	String	TRUE or FALSE to indicate whether the job log file is queued for printing when the job is complete.
JOB_LOWERCASE[1]	String	TRUE or FALSE to indicate whether the job is to be printed on a printer that can print both uppercase and lowercase letters.
JOB_NAME[1]	String	The name of the specified job.
JOB_NOTIFY[1]	String	TRUE or FALSE to indicate whether a message is broadcast to a terminal when a job completes or aborts.
JOB_PENDING	String	TRUE or FALSE to indicate whether the job is pending.
JOB_PID	String	The process identification (PID) number of the executing batch job.
JOB_REFUSED	String	TRUE or FALSE to indicate whether the job was refused by the symbiont and is waiting for the symbiont to accept it for processing.
JOB_RESET_MODULES	String	The names of the text modules that are to be extracted from the device control library and copied to the printer before each job in the specified queue is printed. This item code is meaningful only for output execution queues.
JOB_RESTART[1]	String	TRUE or FALSE to indicate whether the job will restart after a system failure or can be requeued during execution.
JOB_RETAINED	String	TRUE or FALSE to indicate whether the job has completed but is being retained in the queue.
JOB_RETENTION	String	TRUE or FALSE to indicate whether the user requested that the job be retained indefinitely in the queue regardless of the job's completion status.
JOB_RETENTION_TIME[1]	String	Returns the system time until which the user requested the job be retained in the queue. The system time may be expressed in either absolute or delta time format.
JOB_SIZE	Integer	The total number of blocks in the specified print job.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
JOB_SIZE_MAXIMUM	Integer	The maximum number of blocks that a print job initiated from the specified queue can contain. This item code is applicable only to output execution queues.
JOB_SIZE_MINIMUM	Integer	The minimum number of blocks that a print job initiated from the specified queue can contain. This item code is applicable only to output execution queues.
JOB_STALLED	String	TRUE or FALSE to indicate whether the specified job is stalled because the physical device on which the job is printing is stalled.
JOB_STARTING	String	TRUE or FALSE to indicate whether the job controller is starting to process the job and has begun communicating with an output symbiont or a job controller on another node.
JOB_STATUS	Integer	The specified job's status flags. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of JOB_STATUS: JOB_ABORTING JOB_EXECUTING JOB_HOLDING JOB_INACCESSIBLE JOB_PENDING JOB_REFUSED JOB_RETAINED JOB_STALLED JOB_STARTING JOB_SUSPENDED JOB_TIMED_RELEASE
JOB_SUSPENDED	String	TRUE or FALSE to indicate whether the job is suspended.
JOB_TIMED_RELEASE	String	TRUE or FALSE to indicate whether the job is waiting for a specified time to execute.
JOB_WSDEFAULT[1]	String	TRUE or FALSE to indicate whether a default working set size is specified for the job.
JOB_WSEXTENT[1]	String	TRUE or FALSE to indicate whether a working set extent is specified for the job.
JOB_WSQUOTA[1]	String	TRUE or FALSE to indicate whether a working set quota is specified for the job.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
LAST_PAGE[1]	Integer	The page number at which the printing of the specified file should end. This item code is applicable only to output execution queues.
LIBRARY_SPECIFICATION	String	The name of the device control library for the specified queue. The library specification assumes the device and directory name SYS$LIBRARY and a file type of TLB. This item code is meaningful only for output execution queues.
LOG_QUEUE[1]	String	The name of the queue into which the log file produced for the specified batch job is to be entered for printing. This item code is applicable only to batch jobs.
LOG_SPECIFICATION[1]	String	The name of the log file specified for a job. This item code is meaningful only for batch jobs. Use the JOB_LOG_NULL item code to determine whether a log file will be produced.
MANAGER_NAME	String	The queue manager name.
MANAGER_NODES	String	The names of the nodes on which the queue manager may run.
MANAGER_STATUS	Integer	The specified queue manager's status flags. To find the settings of each bit in the field, use one of the following items in place of JOB_STATUS: MANAGER_FAILOVER MANAGER_RUNNING MANAGER_START_PENDING MANAGER_STARTING MANAGER_STOPPED MANAGER_STOPPING
NOTE[1]	String	The note that is to be printed on the job flag and file flag pages of the specified job. This item code is meaningful only for output execution queues.
OPERATOR_REQUEST[1]	String	The message that is to be sent to the queue operator before the specified job begins to execute. This item code is meaningful only for output execution queues.
OWNER_UIC[1]	String	The owner user identification code (UIC) of the specified queue.
PAGE_SETUP_MODULES	String	The names of the text modules to be extracted from the device control library and copied to the printer before each page of the specified form is printed.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
PARAMETER_1 to PARAMETER_8[1]	String	The value of the user-defined parameters that become the value of the DCL symbols P1 to P8 respectively.
PENDING_JOB_BLOCK_ COUNT	Integer	The total number of blocks for all pending jobs in the queue (valid only for output execution queues).
PENDING_JOB_COUNT	Integer	The number of jobs in the queue in a pending state.
PENDING_JOB_REASON	Integer	The reason that the job is in a pending state. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of PENDING_JOB_ REASON: PEND_CHAR_MISMATCH PEND_JOB_SIZE_MAX PEND_JOB_SIZE_MIN PEND_LOWERCASE_MISMATCH PEND_NO_ACCESS PEND_QUEUE_BUSY PEND_QUEUE_STATE PEND_STOCK_MISMATCH
PEND_CHAR_MISMATCH	String	TRUE or FALSE to indicate whether the job requires characteristics that are not available on the execution queue.
PEND_JOB_SIZE_MAX	String	TRUE or FALSE to indicate whether the block size of the job exceeds the upper block limit of the execution queue.
PEND_JOB_SIZE_MIN	String	TRUE or FALSE to indicate whether the block size of the job is less than the lower limit of the execution queue.
PEND_LOWERCASE_ MISMATCH	String	TRUE or FALSE to indicate whether the job requires a lowercase printer.
PEND_NO_ACCESS	String	TRUE or FALSE to indicate whether the owner of the job does not have access to the execution queue.
PEND_QUEUE_BUSY	String	TRUE or FALSE to indicate whether the job is pending because the number of jobs currently executing on the queue equals the job limit for the queue.
PEND_QUEUE_STATE	String	TRUE or FALSE to indicate whether the job is pending because the execution queue is not in a running open state.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
PEND_STOCK_MISMATCH	String	TRUE or FALSE to indicate whether the stock type required by the job's form does not match the stock type of the form mounted on the execution queue.
PRIORITY[1]	Integer	The scheduling priority of the specified job.
PROCESSOR	String	The name of the symbiont image that executes print jobs initiated from the specified queue.
PROTECTION[1]	String	The specified queue's protection mask.
QUEUE_ACL_SPECIFIED	String	TRUE or FALSE to indicate whether an access control list has been specified for the queue.
QUEUE_ALIGNING	String	TRUE or FALSE to indicate whether the queue is currently printing alignment pages. A queue prints alignment pages when it is restarted from a paused state by using the command START /QUEUE/ALIGN.
QUEUE_AUTOSTART	String	TRUE or FALSE if the specified queue has been designated as an AUTOSTART queue.
QUEUE_AUTOSTART_ INACTIVE	String	TRUE or FALSE if the queue is an autostart queue that will not be automatically started. If TRUE, a START/QUEUE or INIT/QUEUE /START command must be issued to restart the queue.
QUEUE_AVAILABLE	String	TRUE or FALSE if the queue is processing one or more jobs but is capable of processing one or more additional jobs.
QUEUE_BATCH	String	TRUE or FALSE to indicate whether the queue is a batch queue or a generic batch queue.
QUEUE_BUSY	String	TRUE or FALSE if the number of jobs currently executing on the queue equals the job limit for the queue.
QUEUE_CLOSED	String	TRUE or FALSE to indicate whether the queue is closed and will not accept new jobs until the queue is put in an open state.
QUEUE_CPU_DEFAULT	String	TRUE or FALSE to indicate whether a default CPU time limit has been specified for all jobs in the queue.
QUEUE_CPU_LIMIT	String	TRUE or FALSE to indicate whether a maximum CPU time limit has been specified for all jobs in the queue.
QUEUE_DESCRIPTION	String	The description of the queue that was defined by using the /DESCRIPTION qualifier with the INITIALIZE/QUEUE command.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_DIRECTORY	String	The device and directory specification of the queue database directory for the queue manager.
QUEUE_FILE_BURST	String	TRUE or FALSE to indicate whether burst and flag pages precede each file in each job initiated from the queue.
QUEUE_FILE_BURST_ONE	String	TRUE or FALSE to indicate whether burst and flag pages precede only the first copy of the first file in each job initiated from the queue.
QUEUE_FILE_FLAG	String	TRUE or FALSE to indicate whether a flag page precedes each file in each job initiated from the queue.
QUEUE_FILE_FLAG_ONE	String	TRUE or FALSE to indicate whether a flag page precedes only the first copy of the first file in each job initiated from the queue.
QUEUE_FILE_PAGINATE	String	TRUE or FALSE to indicate whether the output symbiont paginates output for each job initiated from this queue. The output symbiont paginates output by inserting a form feed whenever output reaches the bottom margin of the form.
QUEUE_FILE_TRAILER	String	TRUE or FALSE to indicate whether a trailer page follows each file in each job initiated from the queue.
QUEUE_FILE_TRAILER_ONE	String	TRUE or FALSE to indicate whether a trailer page follows only the last copy of the last file in each job initiated from the queue.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_FLAGS	Integer	The processing options that have been selected for the specified queue. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of QUEUE_FLAGS: QUEUE_ACL_SPECIFIED QUEUE_AUTOSTART QUEUE_BATCH QUEUE_CPU_DEFAULT QUEUE_CPU_LIMIT QUEUE_FILE_BURST QUEUE_FILE_BURST_ONE QUEUE_FILE_FLAG QUEUE_FILE_FLAG_ONE QUEUE_FILE_PAGINATE QUEUE_FILE_TRAILER QUEUE_FILE_TRAILER_ONE QUEUE_GENERIC QUEUE_GENERIC_SELECTION QUEUE_JOB_BURST QUEUE_JOB_FLAG QUEUE_JOB_SIZE_SCHED QUEUE_JOB_TRAILER QUEUE_NO_INITIAL_FF QUEUE_PRINTER QUEUE_RECORD_BLOCKING QUEUE_RETAIN_ALL QUEUE_RETAIN_ERROR QUEUE_SWAP QUEUE_TERMINAL QUEUE_WSDEFAULT QUEUE_WSEXTENT QUEUE_WSQUOTA
QUEUE_GENERIC	String	TRUE or FALSE to indicate whether the queue is a generic queue.
QUEUE_GENERIC_ SELECTION	String	TRUE or FALSE to indicate whether the queue is an execution queue that can accept work from a generic queue.
QUEUE_IDLE	String	TRUE or FALSE to indicate whether the queue is not processing any jobs and is capable of doing so or whether the generic queue is capable of feeding executor queues.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_JOB_BURST	String	TRUE or FALSE to indicate whether burst and flag pages precede each job initiated from the queue.
QUEUE_JOB_FLAG	String	TRUE or FALSE to indicate whether a flag page precedes each job initiated from the queue.
QUEUE_JOB_SIZE_SCHED	String	TRUE or FALSE to indicate whether jobs initiated from the queue are scheduled according to size with the smallest job of a given priority processed first. (Meaningful only for output queues.)
QUEUE_JOB_TRAILER	String	TRUE or FALSE to indicate whether a trailer page follows each job initiated from the queue.
QUEUE_LOWERCASE	String	TRUE or FALSE to indicate whether queue is associated with a printer that can print both uppercase and lowercase characters.
QUEUE_NAME[1]	String	The name of the specified queue or the name of the queue that contains the specified job.
QUEUE_PAUSED	String	TRUE or FALSE to indicate whether execution of all current jobs in the queue is temporarily halted.
QUEUE_PAUSING	String	TRUE or FALSE to indicate whether the queue is temporarily halting execution. Currently executing jobs are completing; temporarily, no new jobs can begin executing.
QUEUE_PRINTER	String	TRUE or FALSE to indicate whether the queue is a printer queue.
QUEUE_RECORD_BLOCKING	String	TRUE or FALSE to indicate whether the symbiont is permitted to concatenate, or block together, the output records it sends to the output device.
QUEUE_REMOTE	String	TRUE or FALSE to indicate whether the queue is assigned to a physical device that is not connected to the local node.
QUEUE_RESETTING	String	TRUE or FALSE to indicate whether the queue is resetting and stopping.
QUEUE_RESUMING	String	TRUE or FALSE to indicate whether the queue is restarting after pausing.
QUEUE_RETAIN_ALL	String	TRUE or FALSE to indicate whether all jobs initiated from the queue remain in the queue after they finish executing. Completed jobs are marked with a completion status.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_RETAIN_ERROR	String	TRUE or FALSE to indicate whether only jobs that do not complete successfully are retained in the queue.
QUEUE_SERVER	String	TRUE or FALSE to indicate whether queue processing is directed to a server symbiont.
QUEUE_STALLED	String	TRUE or FALSE to indicate whether the physical device to which the queue is assigned is stalled; that is, the device has not completed the last I/O request submitted to it.
QUEUE_STARTING	String	TRUE or FALSE to indicate whether the queue is starting.
QUEUE_STATUS	Integer	The specified queue's status flags. The integer represents a bit field. To find the settings of each bit in the field, use one of the following items in place of QUEUE_STATUS: QUEUE_ALIGNING QUEUE_AUTOSTART QUEUE_AUTOSTART_INACTIVE QUEUE_AVAILABLE QUEUE_BUSY QUEUE_CLOSED QUEUE_IDLE QUEUE_LOWERCASE QUEUE_PAUSED QUEUE_PAUSING QUEUE_REMOTE QUEUE_RESETTING QUEUE_RESUMING QUEUE_SERVER QUEUE_STALLED QUEUE_STARTING QUEUE_STOP_PENDING QUEUE_STOPPED QUEUE_STOPPING QUEUE_UNAVAILABLE
QUEUE_STOP_PENDING	String	TRUE or FALSE if queue will be stopped when jobs currently in progress have completed.
QUEUE_STOPPED	String	TRUE or FALSE to indicate whether the queue is stopped.
QUEUE_STOPPING	String	TRUE or FALSE to indicate whether the queue is stopping.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
QUEUE_SWAP	String	TRUE or FALSE to indicate whether jobs initiated from the queue can be swapped.
QUEUE_TERMINAL	String	TRUE or FALSE to indicate whether the queue is a terminal queue.
QUEUE_UNAVAILABLE	String	TRUE or FALSE to indicate whether the physical device to which queue is assigned is not available.
QUEUE_WSDEFAULT	String	TRUE or FALSE to indicate whether a default working set size is specified for each job initiated from the queue.
QUEUE_WSEXTENT	String	TRUE or FALSE to indicate whether a working set extent is specified for each job initiated from the queue.
QUEUE_WSQUOTA	String	TRUE or FALSE to indicate whether a working set quota is specified for each job initiated from the queue.
REQUEUE_QUEUE_NAME[1]	String	The name of the queue to which the specified job is reassigned.
RESTART_QUEUE_NAME[1]	String	The name of the queue in which the job will be placed if the job is restarted.
RETAINED_JOB_COUNT	Integer	The number of jobs in the queue retained after successful completion plus those retained on error.
SECURITY_INACCESSIBLE	String	TRUE or FALSE to indicate whether the user has read access to the specified queue.
SCSNODE_NAME	String	The 6-byte name of the VAX node on which jobs initiated from the specified queue execute. The node name matches the value of the system parameter SCSNODE for the target node.
SUBMISSION_TIME[1]	String	The time at which the specified job was submitted to the queue.
TIMED_RELEASE_JOB_ COUNT	Integer	The number of jobs in the queue on hold until a specified time.
UIC[1]	String	The user identification code (UIC) of the owner of the specified job.
USERNAME[1]	String	The user name of the owner of the specified job.
WSDEFAULT[1]	Integer	The default working set size specified for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

(continued on next page)

Table DCLI–11 (Cont.) F$GETQUI Items

Item	Return Type	Information Returned
WSEXTENT[1]	Integer	The working set extent specified for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.
WSQUOTA[1]	Integer	The working set quota for the specified job or queue. This value is meaningful only for batch jobs and execution and output queues.

[1]Requires Read (R) access if used with one of the function codes: DISPLAY_ENTRY, DISPLAY_JOB, or DISPLAY_FILE.

Examples

1.
```
$  BLOCKS = F$GETQUI("DISPLAY_ENTRY" ,"JOB_SIZE", 1347)
```

In this example, the F$GETQUI lexical function is used to obtain the size in blocks of print job 1347. The value returned reflects the total number of blocks occupied by the files associated with the job.

2.
```
$ IF F$GETQUI("DISPLAY_QUEUE", "QUEUE_STOPPED", "VAX1_BATCH") .EQS.
"TRUE" THEN GOTO 500
```

In this example, the F$GETQUI lexical function is used to return a value of TRUE or FALSE depending on whether the queue VAX1_BATCH is in a stopped state. If VAX1_BATCH is not in the system, F$GETQUI returns a null string ("").

3.
```
! This command procedure shows all queues and the jobs in them.
$  TEMP = F$GETQUI("")
$  QLOOP:
$  QNAME = F$GETQUI("DISPLAY_QUEUE","QUEUE_NAME","*")
$  IF QNAME .EQS. "" THEN EXIT
$  WRITE SYS$OUTPUT ""
$  WRITE SYS$OUTPUT "QUEUE: ", QNAME
$  JLOOP:
$  NOACCESS = F$GETQUI("DISPLAY_JOB","JOB_INACCESSIBLE",,"ALL_JOBS")
$  IF NOACCESS .EQS. "TRUE" THEN GOTO JLOOP
$  IF NOACCESS .EQS. "" THEN GOTO QLOOP
$  JNAME = F$GETQUI("DISPLAY_JOB","JOB_NAME",,"FREEZE_CONTEXT")
$  WRITE SYS$OUTPUT "    JOB:  ", JNAME
$  GOTO JLOOP
```

This sample command procedure displays all the queues in the system and all the jobs to which the user has read access in the system. In the outer loop a wildcard display queue operation is performed. No call is made to establish the right to obtain information about the queue, because all users have implicit read access to queue attributes. Because a wildcard queue name is specified (‘‘*’’), wildcard queue context is maintained across calls to F$GETQUI.

In the inner loop, to obtain information about all jobs, we enter nested wildcard mode from wildcard display queue mode. In this loop, a call is made to establish the right to obtain information about these jobs because users do not have implicit read access to jobs. The FREEZE_CONTEXT keyword is used in the request for a job name to prevent the advance of the wildcard context to the next object. After the job name has been retrieved and displayed, the procedure loops back up for the next job. The context is advanced because the procedure has not

used the FREEZE_CONTEXT keyword. The wildcard queue context is dissolved when the list of matching queues is exhausted. Finally, F$GETQUI returns a null string ("") to denote that no more objects match the specified search criteria.

4.
```
$ THIS_NODE = F$EDIT(F$GETSYI("SCSNODE"),"COLLAPSE")
$ TEMP = F$GETQUI("CANCEL_OPERATION")
$ SET NOON
$LOOP:
$ QUEUE = F$GETQUI("DISPLAY_QUEUE","QUEUE_NAME","*","WILDCARD")
$ IF QUEUE .EQS. "" THEN GOTO ENDLOOP
$ IF THIS_NODE .EQS.-
F$GETQUI("DISPLAY_QUEUE","SCSNODE_NAME","*","WILDCARD,FREEZE_CONTEXT")
$ THEN
$    IF .NOT.-
  F$GETQUI("DISPLAY_QUEUE","QUEUE_AUTOSTART","*","WILDCARD,FREEZE_CONTEXT")-
  THEN START/QUEUE 'QUEUE'
$ ENDIF
$ GOTO LOOP
$ENDLOOP:
$ SET ON
```

This command procedure looks at all queues associated with the local cluster node and starts any queue that is not marked as autostart.

The procedure starts by obtaining the nodename of the local system and clearing the F$GETQUI context. In addition, error handling is turned off for the loop so that, if a queue had been started previously, the resulting error from the START QUEUE command does not abort the command procedure.

Inside the loop, the F$GETQUI function gets the next queue name in the queue list. If the result is empty, then it has reached the end of the list and it exits the loop.

The next IF statement checks to see if the queue runs on the local node. If it does, then the next statement checks to see if the queue is marked as an autostart queue. If that is false, then the queue is started with the start command. The loop is then repeated.

The final command of the procedure restores DCL error handling to the previous setting.

F$GETSYI

Returns status and identification information about the local system (or about a node in the local dual-architecture VMScluster system, if your system is part of a VMScluster).

Format

F$GETSYI(item [,node-name] [,cluster-id])

Return Value

Either an integer or a character string, depending on the item you request.

Arguments

item

Indicates the type of information to be reported about the local node (or about another node in your VMScluster, if your system is part of a VMScluster). Specify the item as a character string expression.

You can specify the items in Table DCLI–12 only for your local node; you cannot specify the node argument with these items. You can specify these items whether or not you are in a VMScluster or VAXcluster system.

You can specify the items in Table DCLI–13 for either your local node or for another node in your VAXcluster. The information in this table is returned for your local node if you do not specify the **node-name** argument; the information is returned for the specified node if you include the **node-name** argument. Your system must be a member of a VMScluster or VAXcluster system in order to specify the items in this table, except for CLUSTER_MEMBER. You can specify CLUSTER_MEMBER whether or not your system is a member of a VMScluster or VAXcluster system.

You can also specify any of the system parameters listed in the *OpenVMS System Management Utilities Reference Manual*. However, you can specify system parameters only for your local node; you cannot specify the **node-name** argument with these items.

node-name

Specifies the node in your VMScluster or VAXcluster system for which information is to be returned. Specify the node as a character string expression. You cannot use the asterisk (*) and the percent sign (%) wildcard characters to specify the **node-name** argument. (This argument can be specified only if your system is part of a VMScluster or VAXcluster.)

You can request information about another node in your VMScluster or VAXcluster system only when you specify an item from Table DCLI–13. If you do not specify a node, the default is the current node.

cluster-id

Specifies the cluster node identification number for which the information is to be returned. (This argument can be specified only if your system is part of a VMScluster or VAXcluster.)

You can request information about another node in your VMScluster or VAXcluster system only when you specify an item from Table DCLI–13. If you do not specify a node, the default is the current node.

To get information for all the nodes in a cluster, use the F$CSID lexical function to obtain each cluster system identification number and use the **cluster-id** argument of F$GETSYI to gather information about each node.

Description

The F$GETSYI lexical function invokes the $GETSYI system service to return status and identification information about the local system (or about a node in the local VMScluster or VAXcluster, if your system is part of a cluster). The F$GETSYI function returns information on the items that can be specified with the $GETSYI system service. For more information on the $GETSYI system service, see the *OpenVMS System Services Reference Manual*.

You can specify the node for which you want information by supplying either the **node-name** or the **cluster-id** argument, but not both.

Table DCLI–12 lists the items you can specify with the F$GETSYI lexical function to get information about your local node. Table DCLI–13 lists the items you can specify to get information about either your local node or another node in your VMScluster or VAXcluster system.

Table DCLI–12 F$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
ACTIVECPU_CNT	Integer	The count of CPUs actively participating in the current boot of a symmetric multiprocessing (SMP) system.
ARCHFLAG	String	Architecture flags for the system.
ARCH_NAME	String	Name of CPU architecture; Alpha for OpenVMS Alpha, VAX for OpenVMS VAX.
ARCH_TYPE	Integer	Type of CPU architecture; 1 for VAX, 2 for Alpha.
AVAILCPU_CNT	Integer	The count of CPUs recognized in the system.
BOOTTIME	String	The time the system was booted.
CHARACTER_ EMULATED	String	TRUE or FALSE to indicate whether the character string instructions are emulated on the CPU.
‡CONSOLE_VERSION	String	Console firmware version.
CONTIG_GBLPAGES	Integer	Total number of free, contiguous global pages.

‡Alpha specific.

(continued on next page)

Table DCLI–12 (Cont.) F$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
†CPU	Integer	On VAX, the processor type, as represented in the processor's system identification (SID) register. For example, the integer 1 represents a VAX–11/780 and the integer 6 represents a VAX 8530, VAX 8550, VAX 8700, or VAX 8800.
‡CPUTYPE	Integer	On Alpha, the processor type, as stored in the hardware restart parameter block (HWRPB). The value of 2 represents a DECchip 21064 processor.
DECIMAL_EMULATED	String	TRUE or FALSE to indicate whether the decimal string instructions are emulated on the CPU.
D_FLOAT_EMULATED	String	TRUE or FALSE to indicate whether the D_floating instructions are emulated on the CPU.
ERRORLOGBUFFERS	Integer	Number of system pages (on VAX) or pagelets (on Alpha) to be used as errorlog buffers.
F_FLOAT_EMULATED	String	TRUE or FALSE to indicate whether the F_floating instructions are emulated on the CPU.
FREE_GBLPAGES	Integer	Current count of free global pages.
FREE_GBLSECTS	Integer	Current count of free global section table entries.
G_FLOAT_EMULATED	String	TRUE or FALSE to indicate whether the G_floating instructions are emulated on the CPU.
‡GH_RSRVPGCNT	Integer	On Alpha, number of pages covered by granularity hints to reserve for use by the INSTALL utility after system startup has completed.
‡ITB_ENTRIES	Integer	On Alpha, number of I-stream translation buffer entries which support granularity hints to be allocated for resident code.
MEMSIZE	Integer	Number of pages of memory in the system configuration.
PAGEFILE_FREE	Integer	Number of free pages in the currently installed paging files.

†VAX specific.

‡Alpha specific.

(continued on next page)

Table DCLI–12 (Cont.) F$GETSYI Items for the Local Node Only

Item	Return Type	Information Returned
PAGEFILE_PAGE	Integer	Number of pages in the currently installed paging files.
‡PALCODE_VERSION	String	Version of the PALCODE (privileged architectural library) on your Alpha system.
QUANTUM	Integer	Maximum amount of processor time a process can receive while other processes are waiting.
SCS_EXISTS	String	TRUE or FALSE to indicate whether the system communication subsystem (SCS) is currently loaded on a VAX node.
SID	Integer	System identification register. On Alpha, returns a value where all fields are zero except the CPU type field, which always contains the value of 256.
SWAPFILE_FREE	Integer	Number of free pages in the currently installed swapping files.
SWAPFILE_PAGE	Integer	Number of pages in the currently installed swapping files.
SYSTEM_RIGHTS	String	The contents of the system rights list on the local system. If you specify a remote system, a null string ("") is returned. This item code returns a list of identifier names separated by commas (,).
‡SYSTYPE	Integer	On Alpha, the family or system hardware platform. For example, the integer 2 represents a DEC 4000, the integer 3 represents a DEC 7000 or DEC 10000, and the integer 4 represents a DEC 3000.
USED_GBLPAGCNT	Integer	Number of pages currently in use in the global page table.
USED_GBLPAGMAX	Integer	Maximum number of pages ever in use in the global page table.
VERSION	String	Version of OpenVMS in use (8-character string filled with trailing blanks).

‡Alpha specific.

Table DCLI–13 F$GETSYI Items for the Local Node or for Other Nodes in the VAXcluster

Item	Return Type	Information Returned
CLUSTER_EVOTES	Integer	Total number of votes in the cluster.
CLUSTER_FSYSID	String	System identification number for first node to boot in the cluster (the founding node). This number is returned as a character string containing a hexadecimal number.
CLUSTER_FTIME	String	The time when the first node in the cluster was booted.
CLUSTER_ MEMBER	String	TRUE or FALSE if the node is a member of the local cluster.
CLUSTER_NODES	Integer	Total number of nodes in the cluster, as an integer.
CLUSTER_ QUORUM	Integer	Total quorum for the cluster.
CLUSTER_VOTES	Integer	Total number of votes in the cluster.
DECNET_ FULLNAME	String	The node name of a DECnet Phase IV system or the node full name of a DECnet/OSI system.
HW_MODEL	Integer	An integer that identifies the node's Alpha or VAX model type. An integer greater than 1023 represents the Alpha operating system and an integer less than or equal to 1023 represents the VAX operating system.
HW_NAME	String	The Alpha or VAX model name.
NODENAME	String	Node name (does not include the following double colon).
NODE_AREA	Integer	The DECnet area for the target node.
NODE_CSID	String	The CSID of the specified node, as a string containing a hexadecimal number. The CSID is a form of system identification.
NODE_EVOTES	Integer	Number of votes allotted to the node.
NODE_HWTYPE	String	Hardware type of the specified node.
NODE_HWVERS	String	Hardware version of the specified node.
NODE_NUMBER	Integer	The DECnet number for the specified node.
NODE_QUORUM	Integer	Quorum that the node has.
NODE_SWINCARN	String	Software incarnation number for the specified node. This number is returned as a string containing a hexadecimal number.

(continued on next page)

Table DCLI–13 (Cont.) F$GETSYI Items for the Local Node or for Other Nodes in the VAXcluster

Item	Return Type	Information Returned
NODE_SWTYPE	String	Type of operating system software used by the specified node.
NODE_SWVERS	String	Software version of the specified node.
NODE_SYSTEMID	String	System identification number for the specified node. This number is returned as a string containing a hexadecimal number.
NODE_VOTES	Integer	Number of votes allotted to the node.
‡REAL_CPUTYPE	String	The actual CPU type of the primary CPU of the system.
VECTOR_ EMULATOR	Boolean	Flag indicating the presence of the VAX vector instruction emulator facility (VVIEF) in the system.
VP_MASK	Integer	Mask indicating which processors in the system have vector coprocessors.
VP_NUMBER	Integer	Number of vector processors in the system.

‡Alpha specific.

Examples

1.
```
$ SYSID = F$GETSYI("SID")
$ SHOW SYMBOL SYSID
  SYSID = 19923201  Hex = 01300101 Octal = 000401
```

This example shows how to use the F$GETSYI function to return the information in the system identification register. Use quotation marks (‘‘ ’’) around the argument SID because it is a string literal. The value returned by F$GETSYI is assigned to the symbol SYSID. Because a node is not specified, information about your current node is returned.

2.
```
$ MEM = F$GETSYI("CLUSTER_MEMBER", "LONDON")
$ SHOW SYMBOL MEM
  MEM = "TRUE"
```

This example uses the F$GETSYI function to determine whether the node LONDON is a member of the local cluster. The return value TRUE indicates that the remote node LONDON is a member of the cluster.

3.
```
$ LIM = F$GETSYI("BJOBLIM")
$ SHOW SYMBOL LIM
  LIM = 16   Hex = 00000010  Octal = 00000000020
```

This example uses the system parameter BJOBLIM as an argument for the F$GETSYI function. This argument returns the batch job limit for the current system.

F$IDENTIFIER

Converts an alphanumeric identifier to its integer equivalent, or converts an integer identifier to its alphanumeric equivalent. An identifier is a name or number that identifies a category of users. The system uses identifiers to determine a user's access to a resource.

Format

F$IDENTIFIER(identifier,conversion-type)

Return Value

An integer value if you are converting an identifier from a name to an integer. The F$IDENTIFIER function returns a string if you are converting an identifier from an integer to a name. If you specify an identifier that is not valid, the F$IDENTIFIER function returns a null string ("") (if you are converting from number to name) or a zero (if you are converting from name to number).

Arguments

identifier

Specifies the identifier to be converted. Specify the identifier as an integer expression if you are converting an integer to a name. Specify the identifier as a character string expression if you are converting a name to an integer.

Any identifier holding the Name Hidden attribute will cause the F$IDENTIFIER to return an error when you do not hold the identifier in question or do not have access to the rights database. For further information on the attribute, see the *OpenVMS Guide to System Security*.

conversion-type

Indicates the type of conversion to be performed. If the **identifier** argument is alphanumeric, specify the **conversion-type** argument as a character string containing ''NAME_TO_NUMBER''. If the **identifier** argument is numeric, specify the **conversion-type** argument as a character string containing ''NUMBER_TO_NAME''.

Examples

1.
```
$ UIC_INT= F$IDENTIFIER("SLOANE","NAME_TO_NUMBER")
$ SHOW SYMBOL UIC_INT
  UIC_INT = 15728665   Hex = 00F00019  Octal = 00074000031
$ UIC = F$FAO("!%U",UIC_INT)
$ SHOW SYMBOL UIC
  UIC = [360,031]
```

This example uses the F$IDENTIFIER to convert the member identifier from the UIC [MANAGERS,SLOANE] to an integer. The F$IDENTIFIER function shows that the member identifier SLOANE is equivalent to the integer 15728665. Note that you must specify the identifier SLOANE using uppercase letters.

To convert this octal number to a standard numeric user identification code (UIC), use the F$FAO function with the !%U directive. (This directive converts a longword to a UIC in named format.) In this example, the member identifier SLOANE is equivalent to the numeric UIC [360,031].

2.
```
$ UIC_INT = (%O31 + (%X10000 * %O360))
$ UIC_NAME = F$IDENTIFIER(UIC_INT,"NUMBER_TO_NAME")
$ SHOW SYMBOL UIC_NAME
  UIC_NAME = "ODONNELL"
```

This example obtains the alphanumeric identifier associated with the numeric UIC [360,031]. First, you must obtain the longword integer that corresponds to the UIC [360,031]. To do this, place the member number into the low-order word. Place the group number into the high-order word. Next, use the F$IDENTIFIER function to return the named identifier associated with the integer.

F$INTEGER

Returns the integer equivalent of the result of the specified expression.

Format

F$INTEGER(expression)

Return Value

An integer value that is equivalent to the specified expression.

Argument

expression
Specifies the expression to be evaluated. Specify either an integer or a character string expression.

If you specify an integer expression, the F$INTEGER function evaluates the expression and returns the result. If you specify a string expression, the F$INTEGER function evaluates the expression, converts the resulting string to an integer, and returns the result.

After evaluating a string expression, the F$INTEGER function converts the result to an integer in the following way. If the resulting string contains characters that form a valid integer, the F$INTEGER function returns the integer value. If the string contains characters that do not form a valid integer, the F$INTEGER function returns the integer 1 if the string begins with T, t, Y, or y. The function returns the integer 0 if the string begins with any other character.

Example

```
$ A = "23"
$ B = F$INTEGER("-9" + A)
$ SHOW SYMBOL B
  B = -923 Hex=FFFFFC65 Octal=176145
```

This example shows how to use the F$INTEGER function to equate a symbol to the integer value returned by the function. In the example, the F$INTEGER function returns the integer equivalent of the string expression (''–9'' + A). First, the F$INTEGER function evaluates the string expression by concatenating the string literal ''–9'' with the string literal ''23''. Note that the value of the symbol A is substituted automatically in a string expression. Also note that the plus sign (+) is a string concatenation operator since both arguments are string literals.

After the string expression is evaluated, the F$INTEGER function converts the resulting character string (''–923'') to an integer, and returns the value –923. This integer value is assigned to the symbol B.

F$LENGTH

Returns the length of the specified character string.

Format

F$LENGTH(string)

Return Value

An integer value for the length of the string.

Argument

string
Specifies the character string whose length is being determined. Specify the string argument as a character string expression.

Example

```
$ MESSAGE = F$MESSAGE(%X1C)
$ SHOW SYMBOL MESSAGE
  MESSAGE = "%SYSTEM-F-EXQUOTA, exceeded quota"
$ STRING_LENGTH = F$LENGTH(MESSAGE)
$ SHOW SYMBOL STRING_LENGTH
  STRING_LENGTH = 33   Hex = 00000021  Octal = 000041
```

The first assignment statement uses the F$MESSAGE function to return the message that corresponds to the hexadecimal value 1C. The message is returned as a character string and is assigned to the symbol MESSAGE.

The F$LENGTH function is then used to return the length of the character string assigned to the symbol MESSAGE. You do not need to use quotation marks (‘‘ ’’) when you use the symbol MESSAGE as an argument for the F$LENGTH function. (Quotation marks are not used around symbols in character string expressions.)

The F$LENGTH function returns the length of the character string and assigns it to the symbol STRING_LENGTH. At the end of the example, the symbol STRING_LENGTH has a value equal to the number of characters in the value of the symbol named MESSAGE, that is, 33.

F$LOCATE

Locates a specified portion of a character string and returns as an integer the offset of the first character. (An offset is the position of a character or a substring relative to the begining of the string. The first character in a string is always offset position 0 from the beginning of the string.) If the substring is not found, F$LOCATE returns the length (the offset of the last character in the character string plus one) of the searched string.

Format

F$LOCATE(substring,string)

Return Value

An integer value representing the offset of the substring argument. An offset is the position of a character or a substring relative to the beginning of the string. The first character in a string is always offset position 0 from the beginning of the string (which always begins at the leftmost character).

If the substring is not found, the F$LOCATE function returns an offset of the last character in the character string plus 1. (This equals the length of the string.)

Arguments

substring
Specifies the character string that you want to locate within the string specified in the **string** argument.

string
Specifies the character string to be edited by F$LOCATE.

Examples

1.
```
$ FILE_SPEC = "MYFILE.DAT;1"
$ NAME_LENGTH = F$LOCATE(".",FILE_SPEC)
```

The F$LOCATE function in this example returns the position of the period (.) in the string with respect to the beginning of the string. The period is in offset position 6, so the value 6 is assigned to the symbol NAME_LENGTH. Note that NAME_LENGTH also equals the length of the file name portion of the file specification MYFILE.DAT, that is, 6.

The substring argument, the period, is specified as a string literal and is therefore enclosed in quotation marks (‘‘ ’’). The **string** argument FILE_SPEC is a symbol, so it should not be placed within quotation marks. It is automatically replaced by its current value during the processing of the function.

2.
```
$ INQUIRE TIME "Enter time"
$ IF F$LOCATE(":",TIME) .EQ. F$LENGTH(TIME) THEN -
  GOTO NO_COLON
```

This section of a command procedure compares the results of the F$LOCATE and F$LENGTH functions to see if they are equal. This technique is commonly used to determine whether a character or substring is contained in a string.

In the example, the INQUIRE command prompts for a time value and assigns the user-supplied time to the symbol TIME. The IF command checks for the presence of a colon (:) in the string entered in response to the prompt. If the value returned by the F$LOCATE function equals the value returned by the F$LENGTH function, the colon is not present. You use the .EQ. operator (rather than .EQS.) because the F$LOCATE and F$LENGTH functions return integer values.

Note that quotation marks are used around the substring argument, the colon, because it is a string literal. However, the symbol TIME does not require quotation marks because it is automatically evaluated as a string expression.

F$MESSAGE

Returns as a character string the facility, severity, identification, and text associated with the specified system status code.

Format

F$MESSAGE(status-code[,message-component-list])

Return Value

A character string containing the system message that corresponds to the argument you specify.
Note that, although each message in the system message file has a numeric value or range of values associated with it, there are many possible numeric values that do not have corresponding messages. If you specify an argument that has no corresponding message, the F$MESSAGE function returns a string containing the NOMSG error message.

For more information on system error messages, see the *OpenVMS System Messages: Companion Guide for Help Message Users*.

Arguments

status-code

Specifies the status code for which you are requesting error message text. You must specify the status code as an integer expression.

message-component-list

Specifies the system message component for which information is to be returned. If this parameter is null or unspecified, then all system message components are returned.

Table DCLI–14 describes the valid system message component keywords:

Table DCLI–14 F$MESSAGE Keywords

Component Keyword	Information returned
FACILITY	Facility name
SEVERITY	Severity level indicator
IDENT	Abbreviation of message text
TEXT	Explanation of message

Note that when the FACILITY, SEVERITY, and IDENT code keywords are specified (individually or in combination), the resulting message code is prefaced with the percent (%) character. The individual parts of the message code are separated by hyphens when multiple code keywords are specified.

When only the TEXT keyword is specified, the resulting text is not prefaced with any character. When the TEXT keyword is specified with the FACILITY, SEVERITY, or IDENT code keyword, the message code is separated from the text by a combination of a comma and a blank (,).

Example

```
$ ERROR_TEXT = F$MESSAGE(%X1C)
$ SHOW SYMBOL ERROR_TEXT
  ERROR_TEXT = "%SYSTEM-F-EXQUOTA, exceeded quota"
```

This example shows how to use the F$MESSAGE function to determine the message associated with the status code %X1C. The F$MESSAGE function returns the message string, which is assigned to the symbol ERROR_TEXT.

```
$ SUBMIT IMPORTANT.COM
$ SYNCHRONIZE /entry='$ENTRY'
$ IF $STATUS THEN EXIT
$!
$ JOB_STATUS = $STATUS
$!
$ IF "%JOBDELETE" .EQS. F$MESSAGE (JOB_STATUS, "IDENT")
$ THEN
       .
           .
           .
$ ELSE
$    IF "%JOBABORT" .EQS. F$MESSAGE (JOB_STATUS, "IDENT")
$    THEN
        .
            .
            .
$    ELSE
$       .
            .
            .
$    ENDIF
$ ENDIF
  .
  .
  .
```

This command procedure submits a batch job and waits for it to complete. Upon successful completion, the procedure exits. If the job completes unsuccessfully, more processing is done based on the termination status of the batch job.

The first command submits the command procedure IMPORTANT.COM. In the second command, the SYNCHRONIZE command tells the procedure to wait for the job to finish. The third command determines if the job completed successfully and, if so, the procedure exits. The next command saves the status in a symbol.

The first IF statement uses F$MESSAGE to determine whether the job was deleted before execution. If so, it does some processing, possibly to resubmit the job or to inform a user via MAIL.

The next IF statement uses F$MESSAGE to determine whether the job was deleted during execution. As a result, some cleanup or human intervention may be required, which would be done in the THEN block.

If neither IF statement was true, then some other unsuccessful status was returned. Other processing, which would be done in the block following the ELSE statement, might be required.

F$MODE

Returns a character string showing the mode in which a process is executing. The F$MODE function has no arguments, but must be followed by parentheses.

Format

F$MODE( )

Return Value

The character string INTERACTIVE for interactive processes. If the process is noninteractive, the character string BATCH, NETWORK or OTHER is returned. Note that the return string always contains uppercase letters.

Arguments

None.

Description

The lexical function F$MODE returns a character string showing the mode in which a process is executing. The F$MODE function has no arguments, but must be followed by parentheses.

The F$MODE function is useful in command procedures that must operate differently when executed interactively and noninteractively. You should include either the F$MODE function or the F$ENVIRONMENT function in your login command file to execute different commands for interactive terminal sessions and noninteractive sessions.

If you do not include the F$MODE function to test whether your login command file is being executed from an interactive process, and the login command file is executed from a noninteractive process (such as a batch job), the process may terminate if the login command file contains commands that are appropriate only for interactive processing.

A command procedure can use the F$MODE function to test whether the procedure is being executed during an interactive terminal session. It can direct the flow of execution according to the results of this test.

Example

```
$ IF F$MODE() .NES. "INTERACTIVE" THEN GOTO NON_INT_DEF
$ INTDEF:         ! Commands for interactive terminal sessions
   .
   .
   .
$ EXIT
$ NON_INT_DEF:         !Commands for noninteractive processes
   .
   .
   .
```

This example shows the beginning of a login.com file that has two sets of initialization commands: one for interactive mode and one for noninteractive mode (including batch and network jobs). The IF command compares the

character string returned by F$MODE with the character string INTERACTIVE; if they are not equal, control branches to the label NON_INT_DEF. If the character strings are equal, the statements following the label INTDEF are executed and the procedure exits before the statements at NON_INT_DEF.

F$PARSE

Parses a file specification and returns either the expanded file specification or the particular file specification field that you request.

Format

F$PARSE(filespec [,default-spec] [,related-spec] [,field] [,parse-type])

Return Value

A character string containing the expanded file specification or the field you specify. If you do not provide a complete file specification for the **filespec** argument, the F$PARSE function supplies defaults in the return string. For more information, see the Description section for this lexical function.
In most cases, the F$PARSE function returns a null string ("") if an error is detected during the parse. For example, a null string is returned if the file specification has incorrect syntax or if a disk or directory does not exist, making the file specification logically incorrect. However, when you specify a field name or the SYNTAX_ONLY parse type, F$PARSE returns the appropriate information.

Arguments

filespec
Specifies a character string containing the file specification to be parsed.

The file specification can contain the asterisk (*) and the percent sign (%) wildcard characters. If you use a wildcard character, the file specification returned by the F$PARSE function contains the wildcard.

default-spec
Specifies a character string containing the default file specification.

The fields in the default file specification are substituted in the output string if a particular field in the **filespec** argument is missing. You can make further substitutions in the **filespec** argument by using the **related-spec** argument.

related-spec
Specifies a character string containing the related file specification.

The fields in the related file specification are substituted in the output string if a particular field is missing from both the **filespec** and **default-spec** arguments.

field
Specifies a character string containing the name of a field in a file specification. Specifying the **field** argument causes the F$PARSE function to return a specific portion of a file specification.

Specify one of the following field names (do not abbreviate):

NODE	Node name
DEVICE	Device name
DIRECTORY	Directory name
NAME	File name

TYPE	File type
VERSION	File version number

parse-type
Specifies the type of parsing to be performed. By default, the F$PARSE function verifies that the directory in the file specification exists on the device in the file specification. However, the existence of the directory is not verified if you provide a **field** argument. Note that the device and directory can be explicitly given in one of the arguments, or can be provided by default.

Also, by default the F$PARSE function translates logical names if they are provided in any of the arguments. The F$PARSE function stops iterative translation when it encounters a logical name with the CONCEALED attribute.

You can change how the F$PARSE function parses a file specification by using one of the following keywords:

NO_CONCEAL	Ignores the ‘‘conceal’’ attribute in the translation of a logical name as part of the file specification; that is, logical name translation does not end when a concealed logical name is encountered.
SYNTAX_ONLY	The syntax of the file specification is checked without verifying that the specified directory exists on the specified device.

Description

The F$PARSE function parses file specifications by using the RMS service $PARSE. For more information on the $PARSE service, see the *OpenVMS Record Management Services Reference Manual*.

When you use the F$PARSE function, you can omit those optional arguments to the right of the last argument you specify. However, you must include commas (,) as placeholders if you omit optional arguments to the left of the last argument you specify.

If you omit the device and directory names in the **filespec** argument, the F$PARSE function supplies defaults, first from the **default-spec** argument and second from the **related-spec** argument. If names are not provided by these arguments, the F$PARSE function uses your current default disk and directory.

If you omit the node name, the file name, the file type, or the version number, the F$PARSE function supplies defaults, first from the **default-spec** argument and second from the **related-spec** argument. (Note that the version number is not picked up from the **related-spec** argument.) If names are not provided by these arguments, the F$PARSE function returns a null specification for these fields.

Examples

1.
```
$ SET DEF DISK2:[FIRST]
$ SPEC = F$PARSE("JAMES.MAR","[ROOT]",,,"SYNTAX_ONLY")
$ SHOW SYMBOL SPEC
  SPEC = "DISK2:[ROOT]JAMES.MAR;"
```

In this example, the F$PARSE function returns the expanded file specification for the file JAMES.MAR. The example uses the SYNTAX_ONLY keyword to request that F$PARSE check the syntax, but should not verify that the [ROOT] directory exists on DISK2.

The default device and directory are DISK2:[FIRST]. Because the directory name [ROOT] is specified as the **default-spec** argument in the assignment statement, it is used as the directory name in the output string. Note that the default device returned in the output string is DISK2, and the default version number for the file is null. You must place quotation marks (‘‘ ’’) around the arguments JAMES.MAR and ROOT because they are string literals.

If you had not specified syntax-only parsing, and [ROOT] were not on DISK2, a null string would have been returned.

2.
```
$ SET DEFAULT DB1:[VARGO]
$ SPEC = F$PARSE("INFO.COM",,,"DIRECTORY")
$ SHOW SYMBOL SPEC
  SPEC = "[VARGO]"
```

In this example the F$PARSE function returns the directory name of the file INFO.COM. Note that because the **default-spec** and **related-spec** arguments are omitted from the argument list, commas (,) must be inserted in their place.

3.
```
$ SPEC= F$PARSE("DENVER::DB1:[PROD]RUN.DAT",,,"TYPE")
$ SHOW SYMBOL SPEC
  SPEC = ".DAT"
```

In this example, the F$PARSE function is used to parse a file specification containing a node name. The F$PARSE function returns the file type .DAT for the file RUN.DAT at the remote node DENVER.

F$PID

Returns a process identification (PID) number and updates the context symbol to point to the current position in the system's process list.

Format

F$PID(context-symbol)

Return Value

A character string containing the PID of a process in the system's list of processes.

Argument

context-symbol
Specifies a symbol that DCL uses to store a pointer into the system's list of processes. The F$PID function uses this pointer to return a PID.

Specify the context symbol by using a symbol. The first time you use the F$PID function in a command procedure, you should use a symbol that is either undefined or equated to the null string ("") or a context symbol that has been created by the F$CONTEXT function.

If the context symbol is undefined or equated to a null string, the F$PID function returns the first PID in the system's process list that it has the privilege to access. That is, if you have GROUP privilege and if the context symbol is null or undefined, the F$PID function returns the PID of the first process in your group. If you have WORLD privilege, the F$PID function returns the PID of the first process in the list. If you have neither GROUP nor WORLD privilege, the F$PID returns the first process that you own. Subsequent calls to F$PID return the rest of the processes on the system you are accessing.

If the context symbol has been created by the F$CONTEXT function, the F$PID function returns the first process name in the system's process list that fits the criteria specified in the F$CONTEXT calls. Subsequent calls to F$PID return only the PIDs of those processes that meet the selection criteria set up by the F$CONTEXT function and that are accessible to your current privileges.

Description

The F$PID function returns a process identification (PID) number and updates the context symbol to point to the current position in the system's process list. You can step through all the processes on a system, or use the lexical function F$CONTEXT to specify selection criteria. The function F$CONTEXT is not required.

The PIDs returned by the F$PID function depend on the privilege of your process. If you have GROUP privilege, the F$PID function returns PIDs of processes in your group. If you have WORLD privilege, the F$PID function returns PIDs of all processes on the system. If you lack GROUP or WORLD privilege, the F$PID function returns only those processes that you own.

The F$CONTEXT function enables the F$PID function to retrieve processes from any node in a dual-architecture VMScluster system.

The first time you use the F$PID function, use a symbol that is either undefined or equated to the null string or to a context symbol that has been created by the F$CONTEXT function. This causes the F$PID function to return the first PID in the system's process list that you have the privilege to access. It also causes the F$PID function to initialize the **context-symbol** argument.

Once the **context-symbol** argument is initialized, each subsequent F$PID returns the next PID in sequence, using the selection criteria set up by the F$CONTEXT function, if any, and updates the context symbol. After the last PID in the process list is returned, the F$PID function returns a null string.

Example

```
$ CONTEXT = ""
$ START:
$     PID = F$PID(CONTEXT)
$     IF PID .EQS. "" THEN EXIT
$     SHOW SYMBOL PID
$     GOTO START
```

This command procedure uses the F$PID function to display a list of PIDs. The assignment statement declares the symbol CONTEXT, which is used as the **context-symbol** argument for the F$PID function. Because CONTEXT is equated to a null string, the F$PID function returns the first PID in the process list that it has the privilege to access.

The PIDs displayed by this command procedure depend on the privilege of your process. When run with GROUP privilege, the PIDs of users in your group are displayed. When run with WORLD privilege, the PIDs of all users on the system are displayed. Without GROUP or WORLD privilege, only those processes that you own are displayed.

F$PRIVILEGE

Returns a string value of either TRUE or FALSE, depending on whether your current process privileges match those specified in the argument. You can specify either the positive or negative version of a privilege.

Format

F$PRIVILEGE(priv-states)

Return Value

A character string containing the value TRUE or FALSE. The F$PRIVILEGE function returns the string FALSE if any one of the privileges in the **priv-states** argument list is false.

Arguments

priv-states
Specifies a character string containing a privilege, or a list of privileges separated by commas (,). For a list of process privileges, see the *OpenVMS Guide to System Security*. Specify any one of the process privileges except [NO]ALL.

Description

Use the F$PRIVILEGE function to identify your current process privileges.

If ''NO'' precedes the privilege, the privilege must be disabled in order for the function to return a value of TRUE. The F$PRIVILEGE function checks each of the keywords in the specified list, and if the result for any one is false, the string FALSE is returned.

Example

```
$ PROCPRIV = F$PRIVILEGE("OPER,GROUP,TMPMBX,NONETMBX")
$ SHOW SYMBOL PROCPRIV
  PROCPRIV = "FALSE"
```

The F$PRIVILEGE function is used to test whether the process has OPER, GROUP, and TMPMBX privileges and if you do not have NETMBX privileges.

The process in this example has OPER (operator), GROUP, TMPMBX (temporary mailbox), and NETMBX (network mailbox) privileges. Therefore, a value of FALSE is returned because the process has NETMBX privilege, but NONETMBX was specified in the priv-states list. Although the Boolean result for the other three keywords is true, the entire expression is declared false because the result for NONETMBX was false.

F$PROCESS

Obtains the current process name string. The F$PROCESS function has no arguments, but must be followed by parentheses.

Format

F$PROCESS( )

Return Value

A character string containing the current process name.

Arguments

None.

Example

```
$ NAME = F$PROCESS()
$ SHOW SYMBOL NAME
  NAME = "MARTIN"
```

In this example, the F$PROCESS function returns the current process name and assigns it to the symbol NAME.

F$SEARCH

Searches a directory file and returns the full file specification for a file you specify.

Format

F$SEARCH(filespec[,stream-id])

Return Value

A character string containing the expanded file specification for the **filespec** argument. If the F$SEARCH function does not find the file in the directory, the function returns a null string ("").

Arguments

filespec

Specifies a character string containing the file specification to be searched for. If the device or directory names are omitted, the defaults from your current default disk and directory are used. The F$SEARCH function does not supply defaults for a file name or type. If the version is omitted, the specification for the file with the highest version number is returned. If the **filespec** argument contains the asterisk (*) or the percent sign (%) wildcard characters, each time F$SEARCH is called, the next file specification that agrees with the **filespec** argument is returned. A null string is returned after the last file specification that agrees with the **filespec** argument.

stream-id

Specifies a positive integer representing the search stream identification number.

The search stream identification number is used to maintain separate search contexts when you use the F$SEARCH function more than once and when you supply different **filespec** arguments. If you use the F$SEARCH function more than once in a command procedure and if you also use different **filespec** arguments, specify **stream-id** arguments to identify each search separately.

If you omit the **stream-id** argument, the F$SEARCH function starts searching at the beginning of the directory file each time you specify a different **filespec** argument.

Description

The lexical function F$SEARCH invokes the RMS service $SEARCH to search a directory file and return the full file specification for a file you specify. The F$SEARCH function allows you to search for files in a directory by using the RMS service $SEARCH. For more information on the $SEARCH routine, see the *OpenVMS Record Management Services Reference Manual*.

You can use the F$SEARCH function in a loop in a command procedure to return file specifications for all files that match a **filespec** argument containing an asterisk (*) or a percent sign (%) wildcard character. Each time the F$SEARCH function is executed, it returns the next file specification that matches the file specification that contains a wildcard character. After the last file specification is returned, the next F$SEARCH call returns a null string. When you use the F$SEARCH function in a loop, you must include an asterisk (*) and the percent sign (%) wildcard characters in the **filespec** argument. Otherwise, the F$SEARCH always returns the same file specification.

Note that you must maintain the context of the search stream in one of the following ways:

- Explicitly, by stating a **stream-id** argument
- Implicitly, by omitting the **stream-id** argument and by using the same **filespec** argument each time you execute the F$SEARCH function

If you do not maintain the context of the search stream, you start a new search at the beginning of the directory file each time you specify a different **filespec** argument.

Note

The lexical function F$SEARCH can return any file that matches the selection criteria you specify, and that exists in the directory at some time between the beginning and the end of the search. Files that are created, renamed, or deleted during the search may or may not be returned.

Examples

1.
```
$ START:
$     FILE = F$SEARCH("SYS$SYSTEM:*.EXE")
$     IF FILE .EQS. "" THEN EXIT
$     SHOW SYMBOL FILE
$     GOTO START
```

This command procedure displays the file specifications of the latest version of all .EXE files in the SYS$SYSTEM directory. (Only the latest version is returned because an asterisk (*) wildcard character is not used as the version number.) The **filespec** argument SYS$SYSTEM:*.EXE is surrounded by quotation marks (‘‘ ’’) because it is a character string expression.

Because no **stream-id** argument is specified, the F$SEARCH function uses a single search stream. Each subsequent F$SEARCH call uses the same **filespec** argument to return the next file specification of an .EXE file from SYS$SYSTEM:. After the latest version of each .EXE file has been displayed, the F$SEARCH function returns a null string ("") and the procedure exits.

2.
```
$ START:
$    COM = F$SEARCH ("*.COM;*",1)
$    DAT = F$SEARCH ("*.DAT;*",2)
$    SHOW SYMBOL COM
$    SHOW SYMBOL DAT
$    IF (COM.EQS. "") .AND. (DAT.EQS. "") THEN EXIT
$    GOTO START
```

This command procedure searches the default disk and directory for both .COM and .DAT files. Note that the **stream-id** argument is specified for each F$SEARCH call so that the context for each search is maintained.

The first F$SEARCH call starts searching from the top of the directory file for a file with a type .COM. When it finds a .COM file, a pointer is set to maintain the search context. When the F$SEARCH function is used the second time, it again starts searching from the top of the directory file for a file with a type .DAT. When the procedure loops back to the label START, the **stream-id** argument allows F$SEARCH to start searching in the

correct place in the directory file. After all versions .COM and .DAT files are returned, the procedure exits.

3.
```
$ FILESPEC = F$SEARCH("TRNTO""SMITH SALLY""::DBA1:[PROD]*.DAT")
$ SHOW SYMBOL FILESPEC
  FILESPEC = "TRNTO"smith password"::DBA1:[PROD]CARS.DAT"
```

This example uses the F$SEARCH function to return a file specification for a file at a remote node. The access control string is enclosed in quotation marks because it is part of a character string expression when it is an argument for the F$SEARCH function. To include quotation marks in a character string expression, you must use two sets of quotation marks.

Note that, when the F$SEARCH function returns a node name containing an access control string, it substitutes the word ‘‘password’’ for the actual user password.

F$SETPRV

Enables or disables specified user privileges. The F$SETPRV function returns a list of keywords indicating user privileges; this list shows the status of the specified privileges before F$SETPRV was executed.

Your process must be authorized to set the specified privilege.

For detailed information on privilege restrictions, see the description of the $SETPRV system service in the *OpenVMS System Services Reference Manual*.

Format

F$SETPRV(priv-states)

Return Value

A character string containing keywords for the current process privileges before they were changed by the F$SETPRV function.

Argument

priv-states
Specifies a character string defining a privilege, or a list of privileges separated by commas (,).

For a list of process privileges, see the *OpenVMS User's Manual*.

Description

The lexical function F$SETPRV invokes the $SETPRV system service to enable or disable specified user privileges. The F$SETPRV function returns a list of keywords indicating user privileges; this list shows the status of the specified privileges before F$SETPRV was executed.

The F$SETPRV function returns keywords for your current privileges, whether or not you are authorized to change the privileges listed in the **priv-states** argument. However, the F$SETPRV function enables or disables only the privileges you are authorized to change.

When you run programs or execute procedures that include the F$SETPRV function, be sure that F$SETPRV restores your process to its proper privileged state. For additional information, refer to the examples that follow.

Examples

1.
```
$ OLDPRIV = F$SETPRV("OPER,NOTMPMBX")
$ SHOW SYMBOL OLDPRIV
  OLDPRIV = "NOOPER,TMPMBX"
```

In this example, the process is authorized to change the OPER (operator) and TMPMBX (temporary mailbox) privileges. The F$SETPRV function enables the OPER privilege and disables the TMPMBX privilege. In addition, the F$SETPRV function returns the keywords NOOPER and TMPMBX, showing the state of these privileges before they were changed.

You must place quotation marks (‘‘ ’’) around the list of privilege keywords because it is a string literal.

2.
```
$ SHOW PROCESS/PRIVILEGE

14-DEC-1994 15:55:09.60   RTA1:             User: SMITH

Process privileges:

Process rights identifiers:
 INTERACTIVE
 LOCAL

$ NEWPRIVS = F$SETPRV("ALL, NOOPER")
$ SHOW SYMBOL NEWPRIVS
  NEWPRIVS = "NOCMKRNL,NOCMEXEC,NOSYSNAM,NOGRPNAM,NOALLSPOOL,
      NODETACH,NODIAGNOSE,NOLOG_IO,NOGROUP,NOACNT,NOPRMCEB,NOPRMMBX,
      NOPSWAPM,NOALTPRI,NOSETPRV,NOTMPMBX,NOWORLD,NOMOUNT,NOOPER,
      NOEXQUOTA,NONETMBX,NOVOLPRO,NOPHY_IO,NOBUGCHK,NOPRMGBL,
      NOSYSGBL,NOPFNMAP,NOSHMEM,NOSYSPRV,NOBYPASS,NOSYSLCK,NOSHARE,
      NOUPGRADE,NODOWNGRADE,NOGRPPRV,NOREADALL,NOSECURITY,OPER"
$ SHOW PROCESS/PRIVILEGE

14-DEC-1994 10:21:18.32   User: METCALFE         Process ID:   00000F24
                              Node: TOKNOW           Process name: "_FTA23:"

Authorized privileges:
 NETMBX    SETPRV    SYSPRV    TMPMBX

Process privileges:
 ACNT                 may suppress accounting messages
 ALLSPOOL             may allocate spooled device
 ALTPRI               may set any priority value
 AUDIT                may direct audit to system security audit log
 BUGCHK               may make bug check log entries
 BYPASS               may bypass all object access controls
 CMEXEC               may change mode to exec
 CMKRNL               may change mode to kernel
 DETACH               may create detached processes
 DIAGNOSE             may diagnose devices
 DOWNGRADE            may downgrade object secrecy
 EXQUOTA              may exceed disk quota
 GROUP                may affect other processes in same group
 GRPNAM               may insert in group logical name table
 GRPPRV               may access group objects via system protection
 IMPORT               may set classification for unlabeled object
 LOG_IO               may do logical i/o
 MOUNT                may execute mount acp function
 NETMBX               may create network device
 OPER                 may perform operator functions
 PFNMAP               may map to specific physical pages
 PHY_IO               may do physical i/o
 PRMCEB               may create permanent common event clusters
 PRMGBL               may create permanent global sections
 PRMMBX               may create permanent mailbox
 PSWAPM               may change process swap mode
 READALL              may read anything as the owner
 SECURITY             may perform security administration functions
 SETPRV               may set any privilege bit
 SHARE                may assign channels to non-shared devices
 SHMEM                may create/delete objects in shared memory
 SYSGBL               may create system wide global sections
 SYSLCK               may lock system wide resources
 SYSNAM               may insert in system logical name table
 SYSPRV               may access objects via system protection
 TMPMBX               may create temporary mailbox
 UPGRADE              may upgrade object integrity
 VOLPRO               may override volume protection
 WORLD                may affect other processes in the world
```

```
Process rights:
 INTERACTIVE
 LOCAL

System rights:
 SYS$NODE_TOKNOW

$ NEWPRIVS = F$SETPRV(NEWPRIVS)
$ SHOW PROCESS/PRIVILEGE

14-DEC-1994 16:05:07.23   RTA1:              User: METCALFE

Process privileges:
 OPER                 operator privilege

Process rights identifiers:
 INTERACTIVE
 LOCAL
```

In this example, the DCL command SHOW PROCESS/PRIVILEGE is used to determine the current process privileges. Note that the process has no privileges enabled.

The F$SETPRV function is then used to process the ALL keyword and enable all privileges recording the previous state of each privilege in the symbol NEWPRIVS. Next, F$SETPRV processes the NOOPER keyword and disables the OPER (operator) privilege, recording the previous state of OPER in NEWPRIVS. Note that the OPER privilege appears in the returned string twice: first as NOOPER and then as OPER.

Entering the command SHOW PROCESS/PRIVILEGE now shows that the current process has all privileges enabled except OPER.

If the returned string is used as the parameter to F$SETPRV, the process has the OPER privilege enabled. This occurs because the OPER command was present twice in the symbol NEWPRIVS. As a result, F$SETPRV looked at the first keyword NOOPER and disabled the privilege. Finally, after processing several other keywords in the NEWPRIVS string, the OPER keyword is presented, allowing F$SETPRV to enable the OPER privilege.

If you are using the ALL or NOALL keywords to save your current privilege environment, Digital recommends that you perform the following procedure to modify the process for a command procedure:

```
$ CURRENT_PRIVS = F$SETPRV("ALL")
$ TEMP = F$SETPRV("NOOPER")
```

If you use this procedure, you can then specify the following command statement at the end of your command procedure so that the original privilege environment is restored:

```
$ TEMP = F$SETPRV(CURRENT_PRIVS)
```

3.
```
$ SAVPRIV = F$SETPRV("NOGROUP")
$ SHOW SYMBOL SAVPRIV
  SAVPRIV = "GROUP"
$ TEST = F$PRIVILEGE("GROUP")
$ SHOW SYMBOL TEST
  TEST = "TRUE"
```

In this example, the process is not authorized to change the GROUP privilege. However, the F$SETPRV function still returns the current setting for the GROUP privilege.

The F$PRIVILEGE function is used to see whether the process has GROUP privilege. The return string, TRUE, indicates that the process has GROUP privilege, even though the F$SETPRV function attempted to disable the privilege.

VAX

4.
```
$ SHOW PROCESS/PRIVILEGE
14-DEC-1994 15:55:09.60   RTA1:             User: METCALFE

Process privileges:
 AUDIT               may direct audit to system security audit log
 DOWNGRADE           may downgrade object secrecy
 IMPORT              may set classification for unlabeled object
 UPDATE
```

These process privileges are VAX specific and are used only in Security Enhancement Service Software (SEVMS) on an OpenVMS VAX system.♦

F$STRING

Returns the string that is equivalent to the specified expression.

Format

F$STRING(expression)

Return Value

A character string equivalent to the specified expression.

Argument

expression
The integer or string expression to be evaluated.

If you specify an integer expression, the F$STRING function evaluates the expression, converts the resulting integer to a string, and returns the result. If you specify a string expression, the F$STRING function evaluates the expression and returns the result.

When converting an integer to a string, the F$STRING function uses decimal representation and omits leading zeros. When converting a negative integer, the F$STRING function places a minus sign at the beginning string representation of the integer.

Example

```
$ A = 5
$ B = F$STRING(-2 + A)
$ SHOW SYMBOL B
  B = "3"
```

The F$STRING function in this example converts the result of the integer expression (–2 + A) to the numeric string, ‘‘3’’. First, the F$STRING function evaluates the expression (–2 + A). Note that 5, the value of symbol A, is automatically substituted when the integer expression is evaluated.

After the integer expression is evaluated, the F$STRING function converts the resulting integer, 3, to the string ‘‘3’’. This string is assigned to the symbol B.

F$TIME

Returns the current date and time in absolute time format.

The F$TIME function has no arguments, but must be followed by parentheses.

Format

F$TIME()

Return Value

A character string containing the current date and time. The returned string has the following fixed, 23-character format:
dd-mmm-yyyy hh:mm:ss.cc

When the current day of the month is any of the values 1 to 9, the first character in the returned string is a blank character. The time portion of the string is always in character position 13, at an offset of 12 characters from the beginning of the string.

Note that you must use the assignment operator (=) to preserve the blank character in the returned string. If you use the string assignment operator (:=), the leading blank is dropped.

Arguments

None.

Example

```
$ OPEN/WRITE OUTFILE DATA.DAT
$ TIME_STAMP = F$TIME()
$ WRITE OUTFILE TIME_STAMP
```

This example shows how to use the F$TIME function to time-stamp a file that you create from a command procedure. OUTFILE is the logical name for the file DATA.DAT, which is opened for writing. The F$TIME function returns the current date and time string, and assigns this string to the symbol TIME_STAMP. The WRITE command writes the date and time string to OUTFILE.

F$TRNLNM

Translates a logical name and returns the equivalence name string or the requested attributes of the logical name specified.

Format

F$TRNLNM(logical-name [,table] [,index] [,mode] [,case] [,item])

Return Value

The equivalence name or attribute of the specified logical name. The return value can be a character string or an integer, depending on the arguments you specify with the F$TRNLNM function. If no match is found, a null string ("") is returned.

Arguments

logical-name

Specifies a character string containing the logical name to be translated.

table

Specifies a character string containing the logical name table or tables that the F$TRNLNM function should search to translate the logical name. The table argument must be a logical name that translates to a logical name table or to a list of table names.

A logical name for a logical name table must be defined in one of the following logical name tables:

- LNM$SYSTEM_DIRECTORY
- LNM$PROCESS_DIRECTORY

Note

If you subsequently create a table using the CREATE/NAME_TABLE command and want to make your private table accessible for F$TRNLNM, you must redefine one of the table logical names to include your private table. To see all the tables that are normally searched by F$TRNLNM, issue the following command:

```
$ SHOW LOGICAL/STRUCTURE LNM$DCL_LOGICAL
```

For more information, see the CREATE/NAME_TABLE amd SHOW LOGICAL commands.

If you do not specify a table, the default value is LNM$DCL_LOGICAL. That is, the F$TRNLNM function searches the tables whose names are equated to the logical name LNM$DCL_LOGICAL. Unless LNM$DCL_LOGICAL has been redefined for your process, the F$TRNLNM function searches the process, job, group, and system logical name tables, in that order, and returns the equivalence name for the first match found.

index

Specifies the number of the equivalence name to be returned if the logical name has more than one translation. The index refers to the equivalence strings in the order the names were listed when the logical name was defined.

The index begins with zero; that is, the first name in a list of equivalence names is referenced by the index zero.

If you do not specify the **index** argument, the default is zero.

mode

Specifies a character string containing one of the following access modes for the translation: USER (default), SUPERVISOR, EXECUTIVE, or KERNEL.

The F$TRNLNM function starts by searching for a logical name created with the access mode specified in the **mode** argument. If it does not find a match, the F$TRNLNM function searches for the name created with each inner access mode and returns the first match found. For example, two logical names can have the same name, but one name can be created with user access mode and the other name with executive access mode. If the **mode** argument is USER, the F$TRNLNM function returns the equivalence string for the user-mode, not the executive-mode, logical name.

case

Specifies the type of case translation to be performed. Specify the case argument as either of the following character strings: CASE_BLIND (default) or CASE_ SENSITIVE.

If the translation is case blind, the F$TRNLNM function first searches for a logical name with characters of the same case as the **logical-name** argument. If no match is found, the F$TRNLNM function searches for an uppercase version of the **logical-name** argument and the logical names it is searching. The result of the first successful translation is returned.

If the translation is case sensitive, the F$TRNLNM function searches only for a logical name with characters of the same case as the **logical-name** argument. If no exact match is found, the F$TRNLNM function returns a null string ("").

item

Specifies a character string containing the type of information that F$TRNLNM should return about the specified logical name. Specify one of the following items:

Item	Return Type	Information Returned
ACCESS_ MODE	String	One of the following access modes associated with the logical name: USER, SUPERVISOR, EXECUTIVE, KERNEL.
CONCEALED	String	TRUE or FALSE to indicate whether the CONCEALED attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created. The CONCEALED attribute is used to create a concealed logical name.

Item	Return Type	Information Returned
CONFINE	String	TRUE or FALSE to indicate whether the logical name is confined. If the logical name is confined (TRUE), then the name is not copied to subprocesses. If the logical name is not confined (FALSE), then the name is copied to subprocesses.
CRELOG	String	TRUE or FALSE to indicate whether the logical name was created with the $CRELOG system service or with the $CRELNM system service, using the CRELOG attribute. If the logical name was created with the $CRELOG system service or with the $CRELNM system service, using the CRELOG attribute, then TRUE is returned. Otherwise, FALSE is returned.
LENGTH	Integer	Length of the equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F$TRNLNM function returns the length of the name specified by the index argument.
MAX_INDEX	Integer	The largest index defined for the logical name. The index shows how many equivalence names are associated with a logical name. The index is zero based; that is, the index zero refers to the first name in a list of equivalence names.
NO_ALIAS	String	TRUE or FALSE to indicate whether the logical name has the NO_ALIAS attribute. The NO_ALIAS attribute means that a logical name must be unique within outer access mode.
TABLE	String	TRUE or FALSE to indicate whether the logical name is the name of a logical name table.
TABLE_NAME	String	Name of the table where the logical name was found.
TERMINAL	String	TRUE or FALSE to indicate whether the TERMINAL attribute was specified with the /TRANSLATION_ATTRIBUTES qualifier when the logical name was created. The TERMINAL attribute indicates that the logical name is not a candidate for iterative translation.
VALUE	String	Default. The equivalence name associated with the specified logical name. If the logical name has more than one equivalence name, the F$TRNLNM function returns the name specified by the index argument.

Description

The lexical function F$TRNLNM uses the $TRNLNM system service to translate a logical name and return the equivalence name string, or the requested attributes of the logical name specified. The translation is not iterative; the equivalence string is not checked to determine whether it is a logical name.

When you use the F$TRNLNM function, you can omit optional arguments that can be used to the right of the last argument you specify. However, you must include commas (,) as placeholders if you omit optional arguments to the left of the last argument that you specify.

You can use the F$TRNLNM function in command procedures to save the current equivalence of a logical name and later restore it. You can also use it to test whether logical names have been assigned.

Examples

1.
```
$ SAVE_DIR = F$TRNLNM("SYS$DISK")+F$DIRECTORY()
    .
    .
    .
$ SET DEFAULT 'SAVE_DIR'
```

   The assignment statement concatenates the values returned by the F$DIRECTORY and F$TRNLNM functions, and assigns the resulting string to the symbol SAVE_DIR. The symbol SAVE_DIR consists of a full device and directory name string.

   The argument SYS$DISK is enclosed in quotation marks (‘‘ ’’) because it is a character string. (The command interpreter treats all arguments that begin with alphabetic characters as symbols or lexical functions, unless the arguments are enclosed in quotation marks.) None of the optional arguments is specified, so the F$TRNLNM function uses the defaults.

   At the end of the command procedure, the original default directory is reset. When you reset the directory, you must place single quotation marks (‘ ’) around the symbol SAVE_DIR to force symbol substitution.

2.
```
$ DEFINE/TABLE=LNM$GROUP TERMINAL 'F$TRNLNM("SYS$OUTPUT")'
```

   This example shows a line from a command procedure that (1) uses the F$TRNLNM function to determine the name of the current output device and (2) creates a group logical name table entry based on the equivalence string.

   You must enclose the argument SYS$OUTPUT in quotation marks because it is a character string.

   Also, in this example you must enclose the F$TRNLNM function in single quotation marks to force the lexical function to be evaluated. Otherwise, the DEFINE command does not automatically evaluate the lexical function.

3.
```
$ RESULT= -
_$ F$TRNLNM("INFILE","LNM$PROCESS",0,"SUPERVISOR",,"NO_ALIAS")
$ SHOW SYMBOL RESULT
  RESULT = "FALSE"
```

In this example, the F$TRNLNM function searches the process logical name table for the logical name INFILE. The function starts the search by looking for the logical name INFILE created in supervisor mode. If no match is found, the function looks for INFILE created in executive mode.

When a match is found, the F$TRNLNM function determines whether the name INFILE was created with the NO_ALIAS attribute. In this case, the NO_ALIAS attribute is not specified.

F$TYPE

Returns the data type of a symbol. The string INTEGER is returned if the symbol is equated to an integer, or if the symbol is equated to a string whose characters form a valid integer.

The string STRING is returned if the symbol is equated to a character string whose characters do not form a valid integer.

If the symbol is undefined, a null string ("") is returned.

Format

F$TYPE(symbol-name)

Return Value

The string INTEGER is returned if the symbol is equated to an integer, or if the symbol is equated to a string whose characters form a valid integer.
If the symbol has been produced by a call to the F$CONTEXT function with a context type of PROCESS or by a call to the F$PID function, the string returned is PROCESS_CONTEXT. A symbol retains this type until F$CONTEXT is called with the symbol and the CANCEL keyword, or until a null string ("") is returned by a call to F$PID.

Similarly, the return value is the string CLUSTER_SYSTEM_CONTEXT for symbols created by the F$CSID function.

If the symbol is a context symbol, then the return value will be one of the types shown in Table DCLI–15.

Table DCLI–15 Context Symbol Types

Symbol Type	Lexical Creating Symbol
PROCESS_CONTEXT	F$PID or F$CONTEXT (with PROCESS context type)
CLUSTER_SYSTEM_CONTEXT	F$CSID

The string STRING is returned if the symbol is equated to a character string whose characters do not form a valid integer or whose type is not a context.

If the symbol is undefined, a null string is returned.

Argument

symbol-name
Specifies the name of the symbol to be evaluated.

Examples

1.
```
$ NUM = "52"
$ TYPE = F$TYPE(NUM)
$ SHOW SYMBOL TYPE
  TYPE = "INTEGER"
```

This example uses the F$TYPE function to determine the data type of the symbol NUM. NUM is equated to the character string ‘‘52’’. Because the

characters in the string form a valid integer, the F$TYPE function returns the string INTEGER.

2.
```
$ NUM = 52
$ TYPE = F$TYPE(NUM)
$ SHOW SYMBOL TYPE
  TYPE = "INTEGER"
```

In this example, the symbol NUM is equated to the integer 52. The F$TYPE function shows that the symbol has an integer data type.

3.
```
$ CHAR = "FIVE"
$ TYPE = F$TYPE(CHAR)
$ SHOW SYMBOL TYPE
  TYPE = "STRING"
```

In this example, the symbol CHAR is equated to the character string FIVE. Because the characters in this string do not form a valid integer, the F$TYPE function shows that the symbol has a string value.

4.
```
$ x = F$CONTEXT("PROCESS",CTX,"USERNAME","SMITH")
$ TYPE = F$TYPE(CTX)
$ SHOW SYMBOL TYPE
  TYPE = "PROCESS_CONTEXT"
$ x = F$CONTEXT("PROCESS",CTX,"CANCEL")
$ TYPE = F$TYPE(CTX)
$ SHOW SYMBOL TYPE
  TYPE = ""
```

In this example, the F$TYPE function returns the string PROCESS_ CONTEXT because the symbol has been produced by a call to the F$CONTEXT function with a context type of PROCESS. The symbol returns this type until F$CONTEXT is called with the symbol and the **selection-item** argument value CANCEL.

F$USER

Returns the current user identification code (UIC) in named format as a character string. The F$USER function has no arguments, but must be followed by parentheses.

Format

F$USER( )

Return Value

A character string containing the current UIC, including brackets ([ ]). The UIC is returned in the format [group-identifier, member-identifier].

Arguments

None.

Example

```
$ UIC = F$USER()
$ SHOW SYMBOL UIC
  UIC = "[GROUP6,JENNIFER]"
```

In this example the F$USER function returns the current user identification code and assigns it to the symbol UIC.

F$VERIFY

Returns an integer value indicating whether the procedure verification setting is currently on or off. If used with arguments, the F$VERIFY function can turn the procedure and image verification settings on or off. You must include the parentheses after the F$VERIFY function whether or not you specify arguments.

Format

F$VERIFY([procedure-value] [,image-value])

Return Value

The integer 0 if the procedure verification setting is off, or the integer 1 if the procedure verification setting is on.

Arguments

procedure-value

Specifies an integer expression with a value of 1 to turn procedure verification on, or a value of 0 to turn procedure verification off.

When procedure verification is on, each DCL command line in the command procedure is displayed on the output device. Procedure verification allows you to verify that each command is executing correctly.

If you use the **procedure-value** argument, the function first returns the current procedure verification setting. Then the command interpreter turns the procedure verification on or off, as specified by the argument.

image-value

Specifies an integer expression with a value of 1 to turn image verification on, or a value of 0 to turn image verification off.

When image verification is on, data lines in the command procedure are displayed on the output device.

Description

The lexical function F$VERIFY returns an integer value indicating whether the procedure verification setting is currently on or off. If used with arguments, the F$VERIFY function can turn the procedure and image verification settings on or off. You must include the parentheses after the F$VERIFY function whether or not you specify arguments.

Using the F$VERIFY function in command procedures allows you to test the current procedure verification setting. For example, a command procedure can save the current procedure verification setting before changing it and then later restore the setting. In addition, you can construct a procedure that does not display (or print) commands, regardless of what the initial state of verification is.

When you use the F$VERIFY function, you can specify zero, one, or two arguments. If you do not specify any arguments, neither of the verification settings is changed. If you specify only the **procedure-value** argument, both procedure and image verification are turned on (if the value is 1) or off (if the value is 0).

If you specify both arguments, procedure and image verification are turned on or off independently. If you specify the **image-value** argument alone, only image verification is turned on or off. If you specify the **image-value** argument alone, you must precede the argument with a comma (,).

You can also use the F$ENVIRONMENT function with VERIFY_PROCEDURE or VERIFY_IMAGE as the argument. With the F$ENVIRONMENT function, you can determine either the procedure or image verification setting; the F$VERIFY function determines only the procedure verification setting.

DCL performs the F$VERIFY function even if it appears after a comment character, if it is enclosed in single quotation marks (‘ ’). This is the only processing that DCL performs within a comment.

Examples

1.
```
$ SAVE_PROC_VERIFY = F$ENVIRONMENT("VERIFY_PROCEDURE")
$ SAVE_IMAGE_VERIFY = F$ENVIRONMENT("VERIFY_IMAGE")
$ SET NOVERIFY
    .
    .
    .
$ TEMP = F$VERIFY(SAVE_PROC_VERIFY, SAVE_IMAGE_VERIFY)
```

This example shows an excerpt from a command procedure. The first assignment statement assigns the current procedure verification setting to the symbol SAVE_PROC_VERIFY. The second assignment statement assigns the current image verification setting to the symbol SAVE_IMAGE_VERIFY.

Then, the SET NOVERIFY command disables procedure and image verification. Later, the F$VERIFY function resets the verification settings, using the original values (equated to the symbols SAVE_PROC_VERIFY and SAVE_IMAGE_VERIFY). The symbol TEMP contains the procedure verification before it is changed with the F$VERIFY function. (In this example the value of TEMP is not used.)

2.
```
$ VERIFY = F$VERIFY(0)
    .
    .
    .
 $ IF VERIFY .EQ. 1 THEN SET VERIFY
```

This example shows an excerpt from a command procedure that uses the F$VERIFY function to save the current procedure verification setting and to turn both procedure and image verification off. At the end of the command procedure, if procedure verification was originally on, both the procedure and image verification are turned on.

LIBRARY

Invokes the Librarian utility, which creates, modifies, or describes an object, macro, help, text, or shareable image library. For a complete description of the Librarian utility, see the *OpenVMS Command Definition, Librarian, and Message Utilities Manual*.

Format

LIBRARY library-filespec [input-filespec[,...]]

LICENSE

Invokes the License Management utility, which manages software licenses on the OpenVMS operating system. For a complete description of the License Management utility, see the *OpenVMS License Management Utility Manual*.

Format

LICENSE subcommand parameter

LINK

Invokes the OpenVMS Linker, which links one or more object modules into a program image and defines execution characteristics of the image. For a complete description of the linker, including more information about the LINK command, see the *OpenVMS Linker Utility Manual*.

Format

LINK filespec[,...]

LOGIN Procedure

Initiates an interactive terminal session.

Format

Ctrl/C

Ctrl/Y

Return

Description

There is no LOGIN command. You signal your intention to access the system by pressing the Return Key, Ctrl/C, or Ctrl/Y, on a terminal not currently in use. The system prompts for your user name and your password (and your secondary password, if you have one) and then validates them.

Specify the optional qualifiers immediately after you type your user name; then press the Return key to get the password prompts.

The login procedure performs the following functions:

- Validates your right to access the system by checking your user name and passwords against the entries in the system's user authorization file (UAF).
- Establishes the default characteristics of your terminal session based on your user name entry in the UAF.
- Executes the command procedure file SYS$SYLOGIN.COM if one exists.
- Executes either the command procedure file named LOGIN.COM if one exists in your default directory, or the command file defined in the UAF, if any.

Some systems are set up with a retry facility for users who are accessing the system from remote or dialup locations. With these systems, when you make a mistake typing your user name or password, the system allows you to reenter the information. To reenter your login information, press the Return key. The system displays the user name prompt again. Now retype your user name and press the Return key to send the information to the system. The system displays the password prompt. (There is both a limit to the number of times you can retry to enter your login information and a time limit between tries.)

Qualifiers

/CLI=command-language-interpreter
Specifies the name of an alternate command language interpreter (CLI) to override the default CLI listed in the UAF. The CLI you specify must be located in SYS$SYSTEM and have the file type .EXE.

If you do not specify a command interpreter by using the /CLI qualifier and you do not have a default CLI listed in the UAF, the system supplies the qualifier /CLI=DCL by default.

/COMMAND[=filespec] (default)
/NOCOMMAND
Controls whether to execute your default login command procedure when you log in. Use the /COMMAND qualifier to specify the name of an alternate login

command procedure. If you specify a file name without a file type, the default file type .COM is used. If you specify the /COMMAND qualifier and omit the file specification, your default login command procedure is executed.

Use the /NOCOMMAND qualifier if you do not want your default login command procedure to be executed.

/DISK=device-name[:]
Specifies the name of a disk device to be associated with the logical device SYS$DISK for the terminal session. This specification overrides the default SYS$DISK device established in the UAF.

/NEW_PASSWORD
Requires that you change the account password before logging in (as if the password had expired). Use this qualifier as a shortcut if you had intended to change your password after login, or if you suspect that your password has been detected.

/TABLES=(command-table[,...])
/TABLES=DCLTABLES (default)
Specifies the name of an alternate CLI table to override the default listed in the UAF. This table name is considered a file specification. The default device and directory is SYS$SHARE and the default file type is EXE.

If a logical name is used, the table name specification must be defined in the system logical name table.

If the /CLI qualifier is set to DCL or MCR, the /TABLES qualifier defaults to the correct value. If the /TABLES qualifier is specified without the /CLI qualifier, the CLI specified in the user's UAF will be used.

Examples

1. Ctrl/Y
```
Username:  SMITHSON
Password:  <PASSWORD>
```

   In this example, pressing Ctrl/Y allows you to access the operating system, which immediately prompts for a user name. After validating the user name, the system prompts for the password but does not echo it.

2. Return
```
Username:  HIGGINS/DISK=USER$
Password: <PASSWORD>
Welcome to OpenVMS Alpha (TM) Operating System, Version 6.2 on node LSR3
    Last interactive login on Tuesday, 18-DEC-1995 08:41
    Last non-interactive login on Monday, 19-DEC-1995 15:43
```

   In this Alpha example, the /DISK qualifier requests that the default disk for the terminal session be USER$. The SHOW DEFAULT command shows that USER$ is the default disk.

3. Return
```
Username:  HIGGINS/DISK=USER$
Password: <PASSWORD>
       Welcome to OpenVMS VAX Version 6.2 on node JUPITER
       Last interactive login on Tuesday, 15-DEC-1995 09:16:47.08
       Last non-interactive login on Monday, 14-DEC-1995 17:32:34.27
$ SHOW DEFAULT
  USER$:[HIGGINS]
```

In this VAX example, the /DISK qualifier requests that the default disk for the terminal session be USER$. The SHOW DEFAULT command shows that USER$ is the default disk.

4. Ctrl/C
```
Username: LIZA/CLI=MCR/COMMAND=ALTLOGIN.COM
Password: <PASSWORD>
       Welcome to OpenVMS VAX Version 6.2 on node JUPITER
       Last interactive login on Tuesday, 15-DEC-1995 09:16:47.08
       Last non-interactive login on Monday, 14-DEC-1995 17:32:34.27
>
```

In this example, the /CLI qualifier requests the alternate MCR command interpreter. The /COMMAND qualifier indicates that the login command file ALTLOGIN.COM is to be executed instead of the default login command file.

The right angle-bracket prompt (>) indicates that MCR is active and expects an MCR command.

5. Return
```
Username: XENAKIS
Password: <PASSWORD>
Password: <PASSWORD>
       Welcome to OpenVMS VAX Version 6.2 on node JUPITER
       Last interactive login on Tuesday, 15-DEC-1995 09:16:47.08
       Last non-interactive login on Monday, 14-DEC-1995 17:32:34.27
$
```

In this example, the second password prompt indicates that the user has a secondary password, which must be entered to access the system.

6. Return
```
Username: JONES
Password: <PASSWORD>
User authorization failure
```
Return
```
Username: JONES
Password: <PASSWORD>
Welcome to OpenVMS Alpha (TM) Operating System, Version 6.2 on node LSR3
        Last interactive login on Tuesday, 15-DEC-1995 09:16:47.08
        Last non-interactive login on Monday, 14-DEC-1995 17:32:34.27
        1 failure since last successful login.
$
```

This example shows the ‘‘User authorization failure’’ message, which indicates that the password has been entered incorrectly. After you successfully log in, a message is displayed showing the number of login failures since your last successful login. This message is displayed only if login failures have occurred.

7. Return

```
Username: JOYCE
Password: <PASSWORD>
Welcome to OpenVMS Alpha (TM) Operating System, Version 6.2 on node LSR3
       Last interactive login on Tuesday, 15-DEC-1995 09:16:47.08
       Last non-interactive login on Monday, 14-DEC-1995 17:32:34.27
       WARNING - Primary password has expired; update immediately.
$
```

This example shows the WARNING message, which indicates that your primary password has expired. You must use the SET PASSWORD command to change your password before logging out, or you will be unable to log in again.

For more information on changing your password, see the description of the SET PASSWORD command in this manual.

8. Return

```
Username: MIHALY/NEW_PASSWORD
Password: <PASSWORD>
Password: <PASSWORD>
       Welcome  to OpenVMS VAX Version 6.2 on node JUPITER
       Last interactive login on Tuesday, 15-DEC-1995 09:16:47.08
       Last non-interactive login on Monday, 14-DEC-1995 17:32:34.27
       Your password has expired; you must set a new password to log in.

          Old password: <PASSWORD>

          New password: <PASSWORD>

          Verification: <PASSWORD>
```

In this example, the user enters the /NEW_PASSWORD qualifer after the user name MIHALY. The system then forces the user to set a new password immediately after login. The prompts are the same as those provided when you enter the DCL command SET PASSWORD from the command line.

LOGOUT

Terminates an interactive terminal session.

Format

LOGOUT

Description

You must use the LOGOUT command to end a terminal session. Under most circumstances, if you turn the power off at your terminal or hang up your telephone connection without using the LOGOUT command, you remain logged in.

When you use the SET HOST command to log in to a remote processor, you generally need to use the LOGOUT command to end the remote session.

Qualifiers

/BRIEF
Prints a brief logout message (process name, date, and time) or a full logout message (a brief message plus accounting statistics).

/FULL
Requests the long form of the logout message. When you specify the /FULL qualifier, the command interpreter displays a summary of accounting information for the terminal session. The default qualifier for a batch job is /FULL.

/HANGUP
/NOHANGUP
Determines, for dialup terminals, whether the phone hangs up whenever you log out. By default, the setting of the /HANGUP qualifier for your terminal port determines whether the line is disconnected. Your system manager determines whether you are permitted to use this qualifier.

Examples

1.
```
$ LOGOUT
  HIGGINS    logged out at 14-DEC-1994 17:48:56.73
```

   In this example, the LOGOUT command uses the default brief message form. No accounting information is displayed.

2.
```
$ LOGOUT/FULL
  HIGGINS    logged out at 14-DEC-1994 14:23:45.30
Accounting information:
Buffered I/O count:       22     Peak working set size:     90
Direct I/O count:         10     Peak virtual size:         69
Page faults:              68     Mounted volumes:            0
Charged CPU time: 0 00:01:30.50  Elapsed time:      0 04:59:02.63
Charged vector CPU time: 0 00:00:21.62
```

   In this example, the LOGOUT command with the /FULL qualifier displays a summary of accounting statistics for the terminal session.

MACRO

Invokes the VAX MACRO assembler to assemble one or more assembly language source files (the default on VAX) or invokes the MACRO-32 compiler for OpenVMS Alpha to compile VAX assembly language source files on Alpha (the default on Alpha).

The /ALPHA qualifier causes the MACRO command to invoke the MACRO-64 assembler if it is installed on Alpha.

The /MIGRATION qualifier causes the MACRO command to invoke the compiler on Alpha.

Format

MACRO filespec[,...]

Parameter

filespec[,...]
Specifies a VAX MACRO assembly language source file to be assembled. If you specify more than one file, separate the file specifications with either commas (,) or plus signs (+). File specifications separated by commas cause the MACRO assembler to produce an object file (and, if indicated, a listing file) for each specified file. File specifications separated by plus signs are concatenated into one input file and produce a single object file (and listing file).

You cannot include the asterisk (*) and the percent sign (%) wildcard characters in a file specification. For each file specification, the MACRO command supplies a default file type .MAR. The MACRO assembler creates output files of one version higher than the highest version existing in the target directory.

Description

The MACRO command invokes the VAX MACRO assembler to assemble one or more assembly language source files.

The qualifiers to the MACRO command serve as either command (global) qualifiers or positional qualifiers. A **command qualifier** affects all the files specified in the MACRO command. A **positional qualifier** affects only the file that it qualifies. All MACRO qualifiers except the /LIBRARY and /UPDATE qualifiers are usable as either command or positional qualifiers. The /LIBRARY and /UPDATE qualifiers are positional qualifiers only.

See the qualifier descriptions for restrictions.

For a complete functional description of the VAX MACRO assembler directives, see the *VAX MACRO and Instruction Set Reference Manual*. For a complete functional description of the MACRO-32 compiler for OpenVMS Alpha, see *Porting VAX MACRO Code from OpenVMS VAX to OpenVMS Alpha*.

Qualifiers

/ALPHA

Alpha

The /ALPHA qualifier cannot be used with the /UPDATE qualifier.

On Alpha, invokes the MACRO-64 assembler for OpenVMS Alpha (if installed on the system). For a description of the assembler, see the *MACRO–64 Assembler for OpenVMS AXP Systems Reference Manual.*♦

/ANALYSIS_DATA[=filespec]
/NOANALYSIS_DATA (default)

Controls whether the assembler creates an analysis data file for the OpenVMS Source Code Analyzer (SCA), and optionally provides the file specification.

By default, the assembler does not create an analysis data file. If you specify the /ANALYSIS_DATA qualifier without a file specification, the assembler creates a file with the same file name as the first input file for the MACRO command. The default file type for analysis data files is .ANA. When you specify the /ANALYSIS_DATA qualifier, you can control the defaults applied to the output file specification by the placement of the qualifier in the command line.

/CROSS_REFERENCE[=(function[,...])]
/NOCROSS_REFERENCE (default)

Controls whether a listing is produced of the locations in the source file where the specified function (or functions) is defined or referenced. If you specify only one function, you can omit the parentheses.

You can specify the following functions:

ALL	Cross-references directives, macros, operation codes, registers, and symbols
DIRECTIVES	Cross-references directives
MACROS	Cross-references macros
OPCODES	Cross-references operation codes
REGISTERS	Cross-references registers
SYMBOLS	Cross-references symbols

Because the assembler writes the cross-references to the listing file, you must specify the /LIST qualifier with the /CROSS_REFERENCE qualifier. If you specify no functions in the /CROSS_REFERENCE qualifier, the assembler assumes the default value of /CROSS_REFERENCE=(MACROS,SYMBOLS). The /NOCROSS_REFERENCE qualifier excludes the cross-reference listing.

/DEBUG[=option]
/NODEBUG (default)

Includes or excludes local symbols in the symbol table or traceback information in the object module. You can replace the /ENABLE and /DISABLE qualifiers with the /DEBUG and /NODEBUG qualifiers when you use the appropriate DEBUG and TRACEBACK options. The /DEBUG or the /NODEBUG qualifier overrides debugging characteristics set with the .ENABLE or .DISABLE assembler directives.

You can specify one or more of the following options:

ALL	Includes in the object module all local symbols in the symbol table, and provides all traceback information for the debugger. This option is equivalent to /ENABLE=(DEBUG,TRACEBACK).
NONE	Makes local symbols and traceback information in the object module unavailable to the debugger. This option is equivalent to /DISABLE=(DEBUG,TRACEBACK).
SYMBOLS	Makes all local symbols in the object module available to the debugger. Makes traceback information unavailable to the debugger. This option is equivalent to /ENABLE=DEBUG and /DISABLE=TRACEBACK together.
TRACEBACK	Makes traceback information in the object module available to the debugger and local symbols unavailable to the debugger. This option is equivalent to /ENABLE=TRACEBACK and /DISABLE=DEBUG together.

If you specify no options to the /DEBUG qualifier, it assumes the default value of /DEBUG=ALL.

/DIAGNOSTICS[=filespec]
/NODIAGNOSTICS (default)

Creates a file containing assembler messages and diagnostic information. If you omit the file specification, the default file name is the same as the source program; the default file type is DIA.

The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

The diagnostics file is reserved for use with Digital layered products, such as the VAX Language-Sensitive Editor (LSE).

/DISABLE=(function[,...])
/NODISABLE

Provides initial settings for the functions disabled by the .DISABLE assembler directive. You can specify one or more of the following functions:

ABSOLUTE	Assembles relative addresses as absolute addresses.
DEBUG	Includes local symbol table information in the object file for use with the debugger.
GLOBAL	Assumes undefined symbols to be external symbols.
SUPPRESSION	Suppresses listing of unreferenced symbols in the symbol table.
TRACEBACK	Provides traceback information to the debugger.
TRUNCATION	Truncates floating-point numbers (if truncation is disabled, numbers are rounded).
VECTOR	Enables the assembler to accept and correctly process vector code.

If you specify only one function, you can omit the parentheses. If you specify no functions in the /DISABLE qualifier, it assumes the default value of /DISABLE=(ABSOLUTE,DEBUG, TRUNCATION, VECTOR). The /NODISABLE qualifier has the same effect as not specifying the /DISABLE qualifier, or negates the effects of any /DISABLE qualifiers specified earlier in the command line.

/ENABLE=(function[,...])
/NOENABLE

Provides initial settings for the functions controlled by the .ENABLE assembler directive. You can specify one or more of the following functions:

ABSOLUTE	Assembles relative addresses as absolute addresses.
DEBUG	Includes local symbol table information in the object file for use with the debugger.
GLOBAL	Assumes undefined symbols to be external symbols.
SUPPRESSION	Suppresses listing of unreferenced symbols in the symbol table.
TRACEBACK	Provides traceback information to the debugger.
TRUNCATION	Truncates floating-point numbers (if truncation is disabled, numbers are rounded).
VECTOR	Enables the assembler to accept and correctly process vector code.

The /NOENABLE qualifier has the same effect as not specifying the /ENABLE qualifier, or negates the effects of any /ENABLE qualifiers specified earlier in the command line. You can specify one or more of the functions listed in the description of the /DISABLE qualifier. If you specify only one function, you can omit the parentheses. If you specify no functions in the /DISABLE qualifier, it assumes the default value of /ENABLE=(GLOBAL,TRACEBACK,SUPPRESSION).

/LIBRARY
/NOLIBRARY

Positional qualifier. The /LIBRARY qualifier cannot be used with the /UPDATE qualifier.

The associated input file to the /LIBRARY qualifier must be a macro library. The default file type is MLB. The /NOLIBRARY qualifier has the same effect as not specifying the /LIBRARY qualifier, or negates the effects of any /LIBRARY qualifiers specified earlier in the command line.

The assembler can search up to 16 libraries, one of which is always STARLET.MLB. This number applies to a particular assembly, not necessarily to a particular MACRO command. If you enter the MACRO command so that more than one source file is assembled, but the source files are assembled *separately*, you can specify up to 16 macro libraries for each separate assembly. More than one macro library in an assembly causes the libraries to be searched in reverse order of their specification.

A macro call in a source program causes the assembler to begin the following sequence of searches:

1. An initial search of the libraries specified with the .LIBRARY directive. The assembler searches these libraries in the reverse order of that in which they were declared.
2. If the macro definition is not found in any of the libraries specified with the .LIBRARY directive, a search of the libraries specified in the MACRO command line (in the reverse order in which they were specified).
3. If the macro definition is not found in any of the libraries specified in the command line, a search of STARLET.MLB.

/LIST[=filespec]
/NOLIST
Creates or omits an output listing, and optionally provides an output file specification for it. The default file type for the listing file is LIS. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

An interactive MACRO command does not produce a listing file by default. The /NOLIST qualifier, present either explicitly or by default, causes errors to be reported on the current output device.

The /LIST qualifier is the default for a MACRO command in a batch job. The /LIST qualifier allows you to control the defaults applied to the output file specification by the placement of the qualifier in the command line. For more information on entering output file qualifiers, see the *OpenVMS User's Manual*.

/MIGRATION
The /MIGRATION qualifier cannot be used with the /UPDATE qualifier.

Alpha

On Alpha, invokes the MACRO-32 compiler for OpenVMS Alpha to compile one or more VAX MACRO assembly language source files into native OpenVMS Alpha object code.

For a complete description of the MACRO-32 compiler for OpenVMS Alpha, see *Porting VAX MACRO Code from OpenVMS VAX to OpenVMS Alpha*.♦

/OBJECT[=filespec]
/NOOBJECT
Creates or omits an object module. It also defines the file specification. By default, the assembler creates an object module with the same file name as the first input file. The default file type for object files is OBJ. The asterisk (*) and the percent sign (%) wildcard characters are not allowed in the file specification.

The /OBJECT qualifier controls the defaults applied to the output file specification by the placement of the qualifier in the command line. For more information on entering output file qualifiers, see the *OpenVMS User's Manual*.

/PREPROCESS_ONLY[=filespec]
/NOPREPROCESS (D)
Causes the assembler to perform only the actions of the preprocessor phase and writes the resulting processed text out to a file.

The filespec defaults to the name of the primary source file. The default file type is .ASM.

/SHOW[=(function[,...])]
/NOSHOW[=(function[,...])]
Provides initial settings for the functions controlled by the assembler directives .SHOW and .NOSHOW.

You can specify one or more of the following functions:

CONDITIONALS	Lists unsatisfied conditional code associated with .IF and .ENDC MACRO directives.
CALLS	Lists macro calls and repeat range expansions.
DEFINITIONS	Lists macro definitions.

EXPANSIONS	Lists macro expansions.
BINARY	Lists binary code generated by the expansion of macro calls.

If you specify more than one function, separate each with a comma and enclose the list in parentheses. If you specify no functions in the /SHOW qualifier, it increments the listing level count; the /NOSHOW qualifier decrements the count in similar circumstances. Because these qualifiers contribute to the listing file, you must also specify the /LIST qualifier when you use them. If you do not specify the /SHOW qualifier, the MACRO command assumes a default of /SHOW=(CONDITIONALS,CALLS,DEFINITIONS). If you specify only one function, you can omit the parentheses.

/UPDATE[=(update-filespec[,...])]
/NOUPDATE
Positional qualifier. The /UPDATE qualifier cannot be used with the /ALPHA, the /LIBRARY, or the /MIGRATION qualifier.

Updates the input file it qualifies by using the SUMSLP batch editor and the specified update file or files. By default, the assembler assumes that the update file has the same file name as the input source file and a file type of UPD. You cannot include the asterisk (*) and the percent sign (%) wildcard characters in the file specifications. If it cannot find a specified update file, the assembler prints an informational message and continues the assembly.

If you specify only one update file, you can omit the parentheses. If you specify more than one update file, the assembler merges the contents into a single list of updates before applying the updates to the source file.

The /NOUPDATE qualifier has the same effect as not specifying the /UPDATE qualifier, or negates any /UPDATE qualifiers specified earlier in the command line. The input source file and update files are not changed by the update operation. The effects of the update appear in the compiled output. If you specify the /LIST qualifier with the /UPDATE qualifier, the assembler writes an audit trail of the changes to the listing file.

Examples

1. `$ MACRO/LIST  CYGNUS, LYRA/OBJECT=LYRAN + MYLIB/LIBRARY`

   In this example, the MACRO command requests two separate assemblies. Using MAR as the default file type, MACRO assembles CYGNUS.MAR to produce CYGNUS.LIS and CYGNUS.OBJ. Then it assembles LYRA.MAR and creates a listing file named LYRA.LIS and an object module named LYRAN.OBJ. The default output file type for a listing is LIS.

   The command requests the search of the MYLIB library file in the current directory for macro definitions.

2. `$ MACRO  ORION`

   MACRO assembles the file ORION.MAR and creates an object file named ORION.OBJ. Executing the command in a batch job causes MACRO to create a listing file named ORION.LIS.

3.
```
$ MACRO  ALPHA/LIST+MYLIB/LIBRARY-
_$ + [TEST]OLDLIB/LIBRARY + []BETA
$ PRINT ALPHA
```

   MACRO concatenates the files ALPHA.MAR and BETA.MAR to produce an object file named ALPHA.OBJ and a listing file named ALPHA.LIS. The command line requests the search of libraries MYLIB.MLB (in the current default directory) and OLDLIB.MLB (in the directory [TEST]) for macro definitions. When macro calls are found in BETA.MAR, MACRO searches the libraries OLDLIB, MYLIB, and the system library STARLET.MLB, in that order, for the definitions.

   The PRINT command prints the listing file ALPHA.LIS.

4.
```
$ MACRO  DELTA+TESTLIB/LIBRARY, ALPHA+MYLIB/LIBRARY
```

   MACRO requests two separate assemblies. MACRO searches TESTLIB.MLB and the system library STARLET.MLB for macro definitions when macro calls are found in DELTA.MAR, and searches MYLIB.MLB and the system library STARLET.MLB for macro definitions when macro calls are found in ALPHA.MAR.

MAIL

Invokes the Mail utility, which is used to send messages to other users of the system. For a complete description of the Mail utility, see the *OpenVMS User's Manual*.

Format

MAIL [filespec] [recipient-name]

MERGE

Invokes the Sort/Merge utility, which combines 2 to 10 similarly sorted input files and creates a single output file. Note that input files to be merged must be in sorted order. For a complete description of the Sort/Merge utility, see the *OpenVMS User's Manual*.

Format

MERGE input-filespec1,input-filespec2[,...] output-filespec

MESSAGE

Invokes the Message utility, which compiles one or more files of message definitions. For a complete description of the Message utility, see the *OpenVMS Command Definition, Librarian, and Message Utilities Manual*.

Format

MESSAGE filespec[,...]

MONITOR

Invokes the Monitor utility, which monitors classes of systemwide performance data at a specified interval. For a complete description of the Monitor utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

MONITOR [/qualifier[,...]] classname[,...] [/qualifier[,...]]

MOUNT

Invokes the Mount utility, which makes a disk or magnetic tape volume available for processing. For a complete description of the Mount utility, see the *OpenVMS System Management Utilities Reference Manual*.

Format

MOUNT device-name[:][,...] [volume-label[,...]] [logical-name[:]]

Index

A

Access control list editor
  See ACL editor
Accounting
  of detached processes, *DCL N-Z*, DCLII–61
  of terminal session, *DCL N-Z*, DCLII–323
ACCOUNTING command, *DCL A-M*, DCLI–14
  See also SET ACCOUNTING command
ACL editor, *DCL A-M*, DCLI–199; *DCL N-Z*, DCLII–209
Activating an autostart queue, *DCL A-M*, DCLI–210, DCLI–269; *DCL N-Z*, DCLII–385
ALLOCATE command, *DCL A-M*, DCLI–15 to DCLI–17
  See also DISMOUNT command
  See also DEASSIGN command, *DCL A-M*, DCLI–111
Allocating devices, *DCL A-M*, DCLI–15
ANALYZE/AUDIT command, *DCL A-M*, DCLI–18
Analyze/Disk_Structure utility, *DCL A-M*, DCLI–20
ANALYZE/ERROR_LOG command, *DCL A-M*, DCLI–21
ANALYZE/IMAGE command, *DCL A-M*, DCLI–22 to DCLI–24
ANALYZE/MEDIA command, *DCL A-M*, DCLI–25
ANALYZE/OBJECT command, *DCL A-M*, DCLI–26 to DCLI–29
ANALYZE/PROCESS_DUMP command, *DCL A-M*, DCLI–30 to DCLI–33
ANALYZE/RMS_FILE command, *DCL A-M*, DCLI–34
ANALYZE/SYSTEM command, *DCL A-M*, DCLI–35
Analyzing
  dump files, *DCL A-M*, DCLI–30
  global symbol tables, *DCL A-M*, DCLI–23
  image file, *DCL A-M*, DCLI–22
  image file fixup section, *DCL A-M*, DCLI–23
  image file patch text records, *DCL A-M*, DCLI–24
  object files, *DCL A-M*, DCLI–26 to DCLI–29
    debugger information records, *DCL A-M*, DCLI–27
    end-of-module records, *DCL A-M*, DCLI–27
    global symbol directory records, *DCL A-M*, DCLI–28
Analyzing
  object files (cont'd)
    link option specification records, *DCL A-M*, DCLI–28
    module header records, *DCL A-M*, DCLI–28
    module traceback records, *DCL A-M*, DCLI–28
    relocation records, *DCL A-M*, DCLI–29
    text, *DCL A-M*, DCLI–29
  object module, *DCL A-M*, DCLI–26
  output converter, *DCL A-M*, DCLI–67
    processing options, *DCL A-M*, DCLI–71
  patch text record, *DCL A-M*, DCLI–24
  restrictions, *DCL A-M*, DCLI–30
  shareable image file, *DCL A-M*, DCLI–22
APPEND command, *DCL A-M*, DCLI–36 to DCLI–40
  See also COPY command
  using with DECwindows compound documents, *DCL A-M*, DCLI–36
Applications
  running locally, *DCL N-Z*, DCLII–120
  running remotely, *DCL N-Z*, DCLII–120
ASSIGN command, *DCL A-M*, DCLI–41 to DCLI–46
  See also DEASSIGN command
Assigning
  logical queue to an execution queue, *DCL A-M*, DCLI–48
  queue name, *DCL A-M*, DCLI–266
  queue options, *DCL A-M*, DCLI–266; *DCL N-Z*, DCLII–385
  symbols interactively, *DCL A-M*, DCLI–281
= (Assignment statement) command, *DCL A-M*, DCLI–1 to DCLI–4
ASSIGN/MERGE command, *DCL A-M*, DCLI–47
ASSIGN/QUEUE command, *DCL A-M*, DCLI–48, DCLI–49
  See also DEASSIGN/QUEUE command
ASTs (asynchronous system traps)
  specifying quota, *DCL N-Z*, DCLII–61
Asynchronous system traps
  See ASTs
ATTACH command, *DCL A-M*, DCLI–50, DCLI–51

Automatic unshelving
controlling, *DCL N-Z*, DCLII–182
determining, *DCL A-M*, DCLI–349; *DCL N-Z*, DCLII–325
Autostart queues, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–385 to DCLII–386
activating, *DCL A-M*, DCLI–210, DCLI–269; *DCL N-Z*, DCLII–385, DCLII–386
creating, *DCL A-M*, DCLI–269
deactivating, *DCL N-Z*, DCLII–414, DCLII–418
designating, *DCL A-M*, DCLI–268; *DCL N-Z*, DCLII–386
disabling, *DCL A-M*, DCLI–182; *DCL N-Z*, DCLII–420
enabling on a node, *DCL A-M*, DCLI–210
failover, *DCL A-M*, DCLI–182, DCLI–210
preparing for a shutdown, *DCL N-Z*, DCLII–421
preventing failover, *DCL A-M*, DCLI–182
restarting after deactivating, *DCL N-Z*, DCLII–414, DCLII–418
starting, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–385
stopping before shutdown, *DCL A-M*, DCLI–182
Availability of queues, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–386

B

BACKUP command, *DCL A-M*, DCLI–52
Bad block data
on disks, *DCL A-M*, DCLI–264
Base priority
establishing for batch jobs, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–192
Batch jobs
defining default working sets, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–197, DCLII–394, DCLII–432
defining maximum CPU time limit, *DCL A-M*, DCLI–286
defining working set extent, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–394, DCLII–433
defining working set quotas, *DCL A-M*, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–394, DCLII–433
deleting files
after processing, *DCL N-Z*, DCLII–427
deleting log files, *DCL A-M*, DCLI–286; *DCL N-Z*, DCLII–428
end of job on cards, *DCL A-M*, DCLI–216
flushing output buffers, *DCL N-Z*, DCLII–173
holding, *DCL A-M*, DCLI–286; *DCL N-Z*, DCLII–428
keeping log files, *DCL N-Z*, DCLII–428
Batch jobs (cont'd)
limiting CPU time of, *DCL A-M*, DCLI–286; *DCL N-Z*, DCLII–428
log files, *DCL N-Z*, DCLII–424
on remote network node, *DCL N-Z*, DCLII–430
passing parameters to, *DCL N-Z*, DCLII–429
password, *DCL N-Z*, DCLII–10
priority, *DCL N-Z*, DCLII–430
saving log file, *DCL A-M*, DCLI–286
stopping process, *DCL N-Z*, DCLII–402
submitting through cards, *DCL A-M*, DCLI–285
synchronizing with process, *DCL N-Z*, DCLII–436
working set
defining default, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–197, DCLII–394, DCLII–432
defining extent for, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–394, DCLII–433
defining quotas for, *DCL A-M*, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–394, DCLII–433
Batch-oriented editor, *DCL A-M*, DCLI–205
Batch queues
See Queues
Block size options
for files, *DCL A-M*, DCLI–178
specifying cluster size on disk, *DCL A-M*, DCLI–258
Byte dumps, *DCL A-M*, DCLI–191

C

CALL command, *DCL A-M*, DCLI–53 to DCLI–56
CANCEL command, *DCL A-M*, DCLI–57 to DCLI–58
Canceling
logical name assignments, *DCL A-M*, DCLI–111
of detached process wakeup request, *DCL N-Z*, DCLII–60
of subprocess wakeup requests, *DCL N-Z*, DCLII–60
Card readers
end of batch job, *DCL A-M*, DCLI–216
Cards
submitting batch job on, *DCL A-M*, DCLI–285
CDU
invoking, *DCL N-Z*, DCLII–102
Central processing units
See CPUs
Changing queue options, *DCL N-Z*, DCLII–385
Character strings
finding in files, *DCL N-Z*, DCLII–70
specifying case for search, *DCL N-Z*, DCLII–70

Character strings (cont'd)
symbol assignment, *DCL A-M*, DCLI–5
CLOSE command, *DCL A-M*, DCLI–59, DCLI–60
Command Definition utility (CDU)
See CDU
Command interpreters
controlling error checking of, *DCL N-Z*, DCLII–172
specifying alternate, *DCL A-M*, DCLI–419
Command procedures
continuing execution of, *DCL A-M*, DCLI–64
controlling error checking in, *DCL N-Z*, DCLII–172
delaying process of, *DCL N-Z*, DCLII–447
displaying command lines of, *DCL N-Z*, DCLII–238
displaying prompts of, *DCL A-M*, DCLI–281
executing, *DCL A-M*, DCLI–9
labels, *DCL A-M*, DCLI–53, DCLI–235, DCLI–237
parameters for, *DCL A-M*, DCLI–9
passing symbols to interactively, *DCL A-M*, DCLI–281
resuming execution of, *DCL A-M*, DCLI–64
stopping and returning to command level 0, *DCL N-Z*, DCLII–402
submitting as batch jobs, *DCL N-Z*, DCLII–424
terminating, *DCL A-M*, DCLI–230
testing expressions, *DCL A-M*, DCLI–253
transferring control within, *DCL A-M*, DCLI–53, DCLI–235, DCLI–237
Command verification
using SET PREFIX, *DCL N-Z*, DCLII–177
Comment characters, *DCL A-M*, DCLI–162
See also DIFFERENCES command
Comment delimiters, *DCL A-M*, DCLI–162
See also DIFFERENCES command
Comparing
characters in records, *DCL A-M*, DCLI–160
files, *DCL A-M*, DCLI–160
Concatenating files, *DCL A-M*, DCLI–36, DCLI–79
CONNECT command, *DCL A-M*, DCLI–61 to DCLI–63
Continuation
of interrupted command procedures, *DCL A-M*, DCLI–64
of interrupted DCL commands, *DCL A-M*, DCLI–64
of interrupted programs, *DCL A-M*, DCLI–64
CONTINUE command, *DCL A-M*, DCLI–64
CONVERT command, *DCL A-M*, DCLI–65
CONVERT/DOCUMENT command, *DCL A-M*, DCLI–66 to DCLI–77
creating an options file, *DCL A-M*, DCLI–71
CONVERT/RECLAIM command, *DCL A-M*, DCLI–78
COPY command, *DCL A-M*, DCLI–79 to DCLI–88
using with DECwindows compound documents, *DCL A-M*, DCLI–80
COPY/FTP command, *DCL A-M*, DCLI–89 to DCLI–90
qualifiers, *DCL A-M*, DCLI–89
Copying directory files, *DCL A-M*, DCLI–81
COPY/RCP command, *DCL A-M*, DCLI–91 to DCLI–92
qualifiers, *DCL A-M*, DCLI–91
CPUs (central processing units)
defining maximum time limit for batch jobs, *DCL A-M*, DCLI–271, DCLI–286; *DCL N-Z*, DCLII–193, DCLII–388
displaying error count for, *DCL N-Z*, DCLII–285
limiting time for batch jobs, *DCL A-M*, DCLI–271; *DCL N-Z*, DCLII–124, DCLII–193, DCLII–388, DCLII–427
time used by current process, *DCL N-Z*, DCLII–351
CREATE command, *DCL A-M*, DCLI–93 to DCLI–95
CREATE/DIRECTORY command, *DCL A-M*, DCLI–96 to DCLI–98
CREATE/FDL command, *DCL A-M*, DCLI–99
CREATE/NAME_TABLE command, *DCL A-M*, DCLI–100 to DCLI–103
CREATE/TERMINAL command, *DCL A-M*, DCLI–104 to DCLI–109
Creating queues, *DCL A-M*, DCLI–266
Ctrl/C key sequence
enabling or disabling, *DCL N-Z*, DCLII–105
Ctrl/O key sequence, *DCL N-Z*, DCLII–438
Ctrl/Q key sequence, *DCL N-Z*, DCLII–438
Ctrl/S key sequence, *DCL N-Z*, DCLII–438
Ctrl/T key sequence
enabling or disabling, *DCL N-Z*, DCLII–105
Ctrl/Y key sequence
enabling or disabling, *DCL N-Z*, DCLII–105

D

Data checks
changing default, *DCL N-Z*, DCLII–241
Data record compaction, *DCL A-M*, DCLI–262
TA90E support, *DCL N-Z*, DCLII–165
Data streams
marking beginning of, *DCL A-M*, DCLI–119
marking end of, *DCL A-M*, DCLI–214
Days
setting primary, *DCL N-Z*, DCLII–107
setting secondary, *DCL N-Z*, DCLII–107
DCL commands
continuing execution of, *DCL A-M*, DCLI–64

DCL commands (cont'd)
marking beginning of input stream, *DCL A-M*, DCLI–119
marking end of input stream, *DCL A-M*, DCLI–214
resuming execution of, *DCL A-M*, DCLI–64
RUN [process] command /ON qualifier, *DCL N-Z*, DCLII–64
SET SECURITY, *DCL N-Z*, DCLII–209
SHOW AUDIT, *DCL N-Z*, DCLII–252
SHOW SECURITY, *DCL N-Z*, DCLII–347
DDIF input converter, *DCL A-M*, DCLI–67
DDIF output converter, *DCL A-M*, DCLI–68
DEALLOCATE command, *DCL A-M*, DCLI–110
See also ALLOCATE command
Deallocating devices, *DCL A-M*, DCLI–110
DEASSIGN command, *DCL A-M*, DCLI–111 to DCLI–114
See also DEFINE command
DEASSIGN/QUEUE command, *DCL A-M*, DCLI–115
DEBUG command, *DCL A-M*, DCLI–116 to DCLI–118
Debugger
analyzing in object files, *DCL A-M*, DCLI–27
information record analysis, *DCL A-M*, DCLI–27
invoking, *DCL A-M*, DCLI–30, DCLI–116
Kept Debugger, *DCL A-M*, DCLI–116
RUN command, *DCL N-Z*, DCLII–56
using with DEPOSIT command, *DCL A-M*, DCLI–155
using with EXAMINE command, *DCL A-M*, DCLI–217
Decimal dump, *DCL A-M*, DCLI–191
DECK command, *DCL A-M*, DCLI–119 to DCLI–121
See also EOD command
DECnet for OpenVMS, *DCL N-Z*, DCLII–140, DCLII–150, DCLII–152
running DECwindows applications across, *DCL N-Z*, DCLII–120
DECterm window
setting application keypad, *DCL A-M*, DCLI–105
DEC Text Processing Utility
See DECTPU
DECTPU
invoking, *DCL A-M*, DCLI–209
DECW$DISPLAY logical name, *DCL N-Z*, DCLII–116, DCLII–278
Default characteristics
modifying terminal, *DCL N-Z*, DCLII–222 to DCLII–235
setting for magnetic tape devices, *DCL N-Z*, DCLII–165
Default devices
displaying, *DCL N-Z*, DCLII–264
Default directories
displaying, *DCL N-Z*, DCLII–264
Default error checking
controlling, *DCL N-Z*, DCLII–172
Default printer
displaying characteristics of, *DCL N-Z*, DCLII–320
Default protection
establishing, *DCL N-Z*, DCLII–191
Default working set
for batch jobs, *DCL A-M*, DCLI–288; *DCL N-Z*, DCLII–197, DCLII–394, DCLII–432
modifying size, *DCL N-Z*, DCLII–245
DEFINE/CHARACTERISTIC command, *DCL A-M*, DCLI–128 to DCLI–129
DEFINE command, *DCL A-M*, DCLI–122 to DCLI–127
See also DEASSIGN command
DEFINE/FORM command, *DCL A-M*, DCLI–130 to DCLI–133
DEFINE/KEY command, *DCL A-M*, DCLI–134 to DCLI–137
Delaying command processing, *DCL N-Z*, DCLII–447
See also Wait states
DELETE/CHARACTERISTIC command, *DCL A-M*, DCLI–142
DELETE command, *DCL A-M*, DCLI–138 to DCLI–141
DELETE/ENTRY command, *DCL A-M*, DCLI–143 to DCLI–145
DELETE/FORM command, *DCL A-M*, DCLI–146
DELETE/INTRUSION_RECORD command, *DCL A-M*, DCLI–147
DELETE/KEY command, *DCL A-M*, DCLI–148 to DCLI–149
DELETE/QUEUE command, *DCL A-M*, DCLI–150 to DCLI–151
DELETE/QUEUE/MANAGER command, *DCL A-M*, DCLI–152
DELETE/SYMBOL command, *DCL A-M*, DCLI–153 to DCLI–154
Deleting
batch job file after processing, *DCL N-Z*, DCLII–427
logical names, *DCL A-M*, DCLI–111
logical name tables, *DCL A-M*, DCLI–111
multiple files, *DCL A-M*, DCLI–138
wakeup request, *DCL N-Z*, DCLII–60
DEPOSIT command, *DCL A-M*, DCLI–155 to DCLI–158
See also EXAMINE command
length qualifiers, *DCL A-M*, DCLI–156
radix qualifiers, *DCL A-M*, DCLI–156

Descriptor dump, *DCL A-M*, DCLI–191
Detached processes
See also Processes
creating, *DCL N-Z*, DCLII–66
image hibernation, *DCL N-Z*, DCLII–60
specifying working set, *DCL N-Z*, DCLII–66
Device driver images
patching, *DCL N-Z*, DCLII–12
Device names
assigning logical names to, *DCL A-M*, DCLI–41, DCLI–122
Devices
access, *DCL A-M*, DCLI–15
allocating, *DCL A-M*, DCLI–15
assigning logical queue name to, *DCL A-M*, DCLI–48
creating, *DCL N-Z*, DCLII–116
deallocating, *DCL A-M*, DCLI–110
dismounting, *DCL A-M*, DCLI–186
displaying
default, *DCL N-Z*, DCLII–264
information on, *DCL N-Z*, DCLII–247
mounted volumes, *DCL N-Z*, DCLII–268
queue entries, *DCL N-Z*, DCLII–280, DCLII–331
status of, *DCL N-Z*, DCLII–266
displaying error count for, *DCL N-Z*, DCLII–285
establishing as spooled, *DCL N-Z*, DCLII–110
establishing operational status for, *DCL N-Z*, DCLII–110
logical name assignment, *DCL A-M*, DCLI–15
magnetic tapes
setting default characteristics for, *DCL N-Z*, DCLII–165
modifying, *DCL N-Z*, DCLII–116
unloading with DISMOUNT command, *DCL A-M*, DCLI–188
virtual, *DCL A-M*, DCLI–220, DCLI–342
DIAGNOSE command, *DCL A-M*, DCLI–159
DIFFERENCES command, *DCL A-M*, DCLI–160 to DCLI–169
comment characters, *DCL A-M*, DCLI–163
comment delimiters, *DCL A-M*, DCLI–162
exit status, *DCL A-M*, DCLI–161
output formats, *DCL A-M*, DCLI–165
DIGITAL Standard Runoff
See DSR
Directories
changing specification, *DCL N-Z*, DCLII–39
copying, *DCL A-M*, DCLI–81
creating, *DCL A-M*, DCLI–96
displaying contents of, *DCL A-M*, DCLI–170
displaying default, *DCL N-Z*, DCLII–264
file version limit
defining at creation, *DCL A-M*, DCLI–97
modifying, *DCL N-Z*, DCLII–113
Directories (cont'd)
modifying number in system space
for Files–11 volume, *DCL N-Z*, DCLII–241
protection
defining at creation, *DCL A-M*, DCLI–97
ready access, *DCL A-M*, DCLI–257
shelved files, *DCL A-M*, DCLI–177
space preallocation on disk, *DCL A-M*, DCLI–259
DIRECTORY command, *DCL A-M*, DCLI–170 to DCLI–181
DISABLE AUTOSTART command, *DCL A-M*, DCLI–182 to DCLI–183
Disabling autostart
on a node, *DCL A-M*, DCLI–182; *DCL N-Z*, DCLII–420
DISCONNECT command, *DCL A-M*, DCLI–184 to DCLI–185
Disk quotas
displaying, *DCL N-Z*, DCLII–345
Disks
allocating mapping pointers, *DCL A-M*, DCLI–264
creating sequential files, *DCL A-M*, DCLI–93
defining shareable volume, *DCL A-M*, DCLI–264
defining structure level, *DCL A-M*, DCLI–264
directory space allocation, *DCL A-M*, DCLI–259
disabling operator status, *DCL N-Z*, DCLII–45
dismounting, *DCL A-M*, DCLI–186
dismounting volume set, *DCL A-M*, DCLI–188
displaying quota, *DCL N-Z*, DCLII–345
enabling operator status, *DCL N-Z*, DCLII–45
establishing operational status for, *DCL N-Z*, DCLII–110
files
comparing, *DCL A-M*, DCLI–160
deleting, *DCL A-M*, DCLI–138
index file placement, *DCL A-M*, DCLI–261
indicating bad block data, *DCL A-M*, DCLI–264
modifying RMS defaults for file operations, *DCL N-Z*, DCLII–205
renaming directory, *DCL N-Z*, DCLII–39
renaming file, *DCL N-Z*, DCLII–39
specifying cluster size, *DCL A-M*, DCLI–258
specifying default file extension size, *DCL A-M*, DCLI–259
specifying density, *DCL A-M*, DCLI–258
specifying faulty areas, *DCL A-M*, DCLI–257
specifying maximum file number, *DCL A-M*, DCLI–262
specifying size for DECram disk, *DCL A-M*, DCLI–264
Disk volumes
initializing, *DCL A-M*, DCLI–256

DISMOUNT command, *DCL A-M*, DCLI–186 to DCLI–189
Dismounting
clusterwide volumes, *DCL A-M*, DCLI–187
disks, *DCL A-M*, DCLI–186
shared devices, *DCL A-M*, DCLI–187
tapes, *DCL A-M*, DCLI–186
Displays
allocated device, *DCL N-Z*, DCLII–267
command procedure, *DCL N-Z*, DCLII–238
current process status, *DCL N-Z*, DCLII–351
date, *DCL N-Z*, DCLII–365
device status, *DCL N-Z*, DCLII–247, DCLII–266
files on current output device, *DCL N-Z*, DCLII–438
names of installed files, *DCL N-Z*, DCLII–267, DCLII–270
names of open files, *DCL N-Z*, DCLII–267
process status, *DCL N-Z*, DCLII–247
system status, *DCL N-Z*, DCLII–247
time, *DCL N-Z*, DCLII–365
working set limit or quota, *DCL N-Z*, DCLII–373
Document conversion
output formats, *DCL A-M*, DCLI–66
Dollar sign ($)
and DECK command, *DCL A-M*, DCLI–119
and EOD command, *DCL A-M*, DCLI–214
and EOJ command, *DCL A-M*, DCLI–216
Domain converter
processing options, *DCL A-M*, DCLI–75
DSR (DIGITAL Standard Runoff), *DCL N-Z*, DCLII–69
invoking, *DCL N-Z*, DCLII–69
text file formatting, *DCL N-Z*, DCLII–69
DTIF input converter, *DCL A-M*, DCLI–68
DTIF output converter, *DCL A-M*, DCLI–69
DUMP command, *DCL A-M*, DCLI–190 to DCLI–198
Dump formats, *DCL A-M*, DCLI–191 to DCLI–195
byte, *DCL A-M*, DCLI–191
decimal, *DCL A-M*, DCLI–191
descriptor, *DCL A-M*, DCLI–191
hexadecimal, *DCL A-M*, DCLI–192
identifier, *DCL A-M*, DCLI–193
longword, *DCL A-M*, DCLI–193
octal, *DCL A-M*, DCLI–193
word, *DCL A-M*, DCLI–195
Dumping
of files, *DCL A-M*, DCLI–190
of volumes, *DCL A-M*, DCLI–190
Dump reading, *DCL A-M*, DCLI–190
Duplicate labels
command interpreter rules for, *DCL A-M*, DCLI–53, DCLI–235, DCLI–237

E

EDIT/ACL command, *DCL A-M*, DCLI–199
EDIT/EDT command, *DCL A-M*, DCLI–200 to DCLI–203
EDIT/FDL command, *DCL A-M*, DCLI–204
Editors
See EDT editor
See EVE
See SUMSLP editor
See TECO editor
EDIT/SUM command, *DCL A-M*, DCLI–205
EDIT/TECO command, *DCL A-M*, DCLI–206 to DCLI–208
EDIT/TPU command, *DCL A-M*, DCLI–209
EDT editor, *DCL A-M*, DCLI–200
ELSE keyword
and IF command, *DCL A-M*, DCLI–253
ENABLE AUTOSTART command, *DCL A-M*, DCLI–210 to DCLI–212
Enabling autostart on a node, *DCL A-M*, DCLI–210
End of batch job on cards, *DCL A-M*, DCLI–216
End of data stream, *DCL A-M*, DCLI–214
See also EOD command
End-of-module record analysis, *DCL A-M*, DCLI–27
ENDSUBROUTINE command, *DCL A-M*, DCLI–54, DCLI–55, DCLI–213
EOD (end of deck)
See EOD command
EOD command, *DCL A-M*, DCLI–214, DCLI–215
and DECK command, *DCL A-M*, DCLI–119
EOF condition, *DCL A-M*, DCLI–214
EOF indicator, *DCL A-M*, DCLI–119
EOJ command, *DCL A-M*, DCLI–216
Equivalence names
assigning to logical names, *DCL A-M*, DCLI–41, DCLI–122
displaying for logical names, *DCL N-Z*, DCLII–366
Error messages
online documentation, *DCL A-M*, DCLI–247
Errors
controlling error checking, *DCL N-Z*, DCLII–172
reporting
for image files, *DCL A-M*, DCLI–22
for object files, *DCL A-M*, DCLI–26
Error streams
defining for created process, *DCL N-Z*, DCLII–59
EVE
invoking, *DCL A-M*, DCLI–209

EXAMINE command, *DCL A-M*, DCLI–217 to DCLI–219
and DEPOSIT command, *DCL A-M*, DCLI–155
length qualifier, *DCL A-M*, DCLI–218
EXCHANGE command, *DCL A-M*, DCLI–220
EXCHANGE/NETWORK command, *DCL A-M*, DCLI–221 to DCLI–229
creating files, *DCL A-M*, DCLI–224
protecting files, *DCL A-M*, DCLI–224
qualifiers, *DCL A-M*, DCLI–225
transferring files, *DCL A-M*, DCLI–223
wildcard character, *DCL A-M*, DCLI–224
Executable images
patching, *DCL N-Z*, DCLII–12
Execute procedure (@) command, *DCL A-M*, DCLI–9 to DCLI–13
Executing
alternate login command procedure, *DCL A-M*, DCLI–419
continuing interrupted command procedures, *DCL A-M*, DCLI–64
continuing interrupted programs, *DCL A-M*, DCLI–64
login command procedure, *DCL A-M*, DCLI–419
SYS$LOGIN, *DCL A-M*, DCLI–419
Execution queues, *DCL A-M*, DCLI–267
designating autostart or nonautostart, *DCL A-M*, DCLI–268
specifying node or node and device, *DCL A-M*, DCLI–269
/EXECUTIVE_MODE qualifier
ASSIGN command, *DCL A-M*, DCLI–42
EXIT command, *DCL A-M*, DCLI–230 to DCLI–233
See also STOP command
Expressions
value tests, *DCL A-M*, DCLI–253
Extensible Versatile Editor
See EVE

F

F$CONTEXT lexical function, *DCL A-M*, DCLI–291, DCLI–294 to DCLI–299
F$CSID lexical function, *DCL A-M*, DCLI–291, DCLI–300, DCLI–301
F$CVSI lexical function, *DCL A-M*, DCLI–292, DCLI–302, DCLI–303
F$CVTIME lexical function, *DCL A-M*, DCLI–292, DCLI–304, DCLI–305
F$CVUI lexical function, *DCL A-M*, DCLI–292, DCLI–306
F$DEVICE lexical function, *DCL A-M*, DCLI–292, DCLI–307, DCLI–308
F$DIRECTORY lexical function, *DCL A-M*, DCLI–292, DCLI–309
F$EDIT lexical function, *DCL A-M*, DCLI–292, DCLI–310, DCLI–311
F$ELEMENT lexical function, *DCL A-M*, DCLI–292, DCLI–312, DCLI–313
F$ENVIRONMENT lexical function, *DCL A-M*, DCLI–292, DCLI–314 to DCLI–316
F$EXTRACT lexical function, *DCL A-M*, DCLI–292, DCLI–317, DCLI–318
F$FAO lexical function, *DCL A-M*, DCLI–292, DCLI–319 to DCLI–325
F$FILE_ATTRIBUTES lexical function, *DCL A-M*, DCLI–292, DCLI–326 to DCLI–328
F$GETDVI lexical function, *DCL A-M*, DCLI–292, DCLI–329 to DCLI–344
F$GETJPI lexical function, *DCL A-M*, DCLI–292, DCLI–345 to DCLI–350
F$GETQUI lexical function, *DCL A-M*, DCLI–292, DCLI–351 to DCLI–373
F$GETSYI lexical function, *DCL A-M*, DCLI–292, DCLI–374 to DCLI–379
F$IDENTIFIER lexical function, *DCL A-M*, DCLI–292, DCLI–380, DCLI–381
F$INTEGER lexical function, *DCL A-M*, DCLI–292, DCLI–382
F$LENGTH lexical function, *DCL A-M*, DCLI–293, DCLI–383
F$LOCATE lexical function, *DCL A-M*, DCLI–293, DCLI–384, DCLI–385
F$LOGICAL lexical function
See F$TRNLNM lexical function
F$MESSAGE lexical function, *DCL A-M*, DCLI–293, DCLI–386, DCLI–387
F$MODE lexical function, *DCL A-M*, DCLI–293, DCLI–388, DCLI–389
F$PARSE lexical function, *DCL A-M*, DCLI–293, DCLI–390 to DCLI–392
F$PID lexical function, *DCL A-M*, DCLI–293, DCLI–393, DCLI–394
F$PRIVILEGE lexical function, *DCL A-M*, DCLI–293, DCLI–395
F$PROCESS lexical function, *DCL A-M*, DCLI–293, DCLI–396
F$SEARCH lexical function, *DCL A-M*, DCLI–293, DCLI–397 to DCLI–399
F$SETPRV lexical function, *DCL A-M*, DCLI–293, DCLI–400 to DCLI–403
F$STRING lexical function, *DCL A-M*, DCLI–293, DCLI–404
F$TIME lexical function, *DCL A-M*, DCLI–293, DCLI–405
F$TRNLNM lexical function, *DCL A-M*, DCLI–293, DCLI–406 to DCLI–410

F$TYPE lexical function, *DCL A-M*, DCLI–411, DCLI–412
F$USER lexical function, *DCL A-M*, DCLI–293, DCLI–413
F$VERIFY lexical function, *DCL A-M*, DCLI–293, DCLI–414, DCLI–415
Failover
  autostart, *DCL A-M*, DCLI–279
  autostart queues, *DCL A-M*, DCLI–182, DCLI–210, DCLI–269; *DCL N-Z*, DCLII–386
  of queue manager, *DCL N-Z*, DCLII–398
  preventing, *DCL A-M*, DCLI–182
False expression
  and IF command, *DCL A-M*, DCLI–253
FAO lexical directives, *DCL A-M*, DCLI–319 to DCLI–325
FHM (file high-water mark), *DCL A-M*, DCLI–261
File expiration date
  specifying retention time values, *DCL N-Z*, DCLII–243
File high-water mark
  See FHM
File images
  analyzing, *DCL A-M*, DCLI–22
  fixup section analyzing, *DCL A-M*, DCLI–23
File names
  changing, *DCL N-Z*, DCLII–39
File objects, *DCL A-M*, DCLI–26 to DCLI–29
  analyzing, *DCL A-M*, DCLI–26
    debugger information records, *DCL A-M*, DCLI–27
    global symbol directory records, *DCL A-M*, DCLI–28
    link option specification records, *DCL A-M*, DCLI–28
    module header records, *DCL A-M*, DCLI–28
    module traceback records, *DCL A-M*, DCLI–28
    relocation records, *DCL A-M*, DCLI–29
    text, *DCL A-M*, DCLI–29
  identifying
    errors, *DCL A-M*, DCLI–26
File protection
  changing default for volume, *DCL N-Z*, DCLII–241
  defining at file creation, *DCL A-M*, DCLI–94
  defining default, *DCL A-M*, DCLI–259
  establishing default, *DCL N-Z*, DCLII–191
  with EXCHANGE/NETWORK command, *DCL A-M*, DCLI–224
Files
  allocating headers, *DCL A-M*, DCLI–260
  appending, *DCL A-M*, DCLI–36
  batch jobs
    deleting after processing, *DCL N-Z*, DCLII–427
Files (cont'd)
  closing, *DCL A-M*, DCLI–59
  closing of queue database, *DCL N-Z*, DCLII–413
  comparing, *DCL A-M*, DCLI–160
  concatenating, *DCL A-M*, DCLI–79
  copying, *DCL A-M*, DCLI–79, DCLI–221
  creating, *DCL A-M*, DCLI–79, DCLI–93, DCLI–221
    with EDT editor, *DCL A-M*, DCLI–200
    with TECO editor, *DCL A-M*, DCLI–206
  creating owner UIC, *DCL A-M*, DCLI–94
  deassigning logical names, *DCL A-M*, DCLI–59
  default extension size on disk, *DCL A-M*, DCLI–259
  deleting, *DCL A-M*, DCLI–138
  displaying
    allocated blocks, *DCL A-M*, DCLI–178
    at terminal, *DCL N-Z*, DCLII–438
    backup date, *DCL A-M*, DCLI–172
    backup time, *DCL A-M*, DCLI–178
    blocks used, *DCL A-M*, DCLI–178
    creation date, *DCL A-M*, DCLI–172
    creation time, *DCL A-M*, DCLI–178
    expiration date, *DCL A-M*, DCLI–172
    expiration time, *DCL A-M*, DCLI–178
    files opened by the system, *DCL N-Z*, DCLII–270
    help, *DCL A-M*, DCLI–239
    latest version, *DCL A-M*, DCLI–179
    modification date, *DCL A-M*, DCLI–172
    modification time, *DCL A-M*, DCLI–178
    names of installed files, *DCL N-Z*, DCLII–267, DCLII–270
    names of open files, *DCL N-Z*, DCLII–267
    on current output device, *DCL N-Z*, DCLII–438
    owner UIC, *DCL A-M*, DCLI–174
    protection, *DCL A-M*, DCLI–175
  dumping, *DCL A-M*, DCLI–190
  editing
    with EDT editor, *DCL A-M*, DCLI–200
    with SUMSLP editor, *DCL A-M*, DCLI–205
    with TECO editor, *DCL A-M*, DCLI–206
  formatting text
    See DSR
  ignoring characters, strings, or records in comparisons, *DCL A-M*, DCLI–163
  list in directory, *DCL A-M*, DCLI–170
  maximum number on disk, *DCL A-M*, DCLI–262
  modifying characteristics, *DCL N-Z*, DCLII–132
  modifying queue entries for, *DCL N-Z*, DCLII–122
  modifying RMS defaults for file operations, *DCL N-Z*, DCLII–205

Files (cont'd)
opening, *DCL N-Z*, DCLII–6
printing, *DCL N-Z*, DCLII–14
purging, *DCL N-Z*, DCLII–28
queue and journal, *DCL N-Z*, DCLII–397
reading records from, *DCL N-Z*, DCLII–32
renaming, *DCL N-Z*, DCLII–39
searching for character string, *DCL N-Z*, DCLII–70
transferring, *DCL A-M*, DCLI–221
unlocking, *DCL N-Z*, DCLII–445
updating with SUMSLP editor, *DCL A-M*, DCLI–205
version limit
defining at directory creation, *DCL A-M*, DCLI–97
writing records to, *DCL N-Z*, DCLII–449
Files–11
initializing disks, *DCL A-M*, DCLI–256
On-Disk Structure Level 1 format, *DCL A-M*, DCLI–256
Files–11 volumes
modifying characteristics of, *DCL N-Z*, DCLII–241
File shareable images
analyzing, *DCL A-M*, DCLI–22
File system requests
responding to, *DCL N-Z*, DCLII–44
File types
changing, *DCL N-Z*, DCLII–39
File version numbers
changing, *DCL N-Z*, DCLII–39
File windows
mapping pointer allocation, *DCL A-M*, DCLI–264
specifying mapping pointers, *DCL N-Z*, DCLII–244
FONT command, *DCL A-M*, DCLI–234
Formatting
of DIFFERENCES output, *DCL A-M*, DCLI–164

G

Generic device names, *DCL A-M*, DCLI–15
Generic queues, *DCL A-M*, DCLI–267
initializing, *DCL A-M*, DCLI–274; *DCL N-Z*, DCLII–390
starting, *DCL A-M*, DCLI–268
stopping, *DCL A-M*, DCLI–268
Global symbol directories
See GSDs
Global symbols, *DCL A-M*, DCLI–1, DCLI–5
Global symbol tables
See GSTs
GOSUB command, *DCL A-M*, DCLI–235, DCLI–236
GOTO command, *DCL A-M*, DCLI–237, DCLI–238
Granularity hint regions (GHR), *DCL N-Z*, DCLII–308
Group logical name tables
canceling entries, *DCL A-M*, DCLI–112
including logical name, *DCL A-M*, DCLI–42, DCLI–123
GSDs (global symbol directories)
analyzing in object files, *DCL A-M*, DCLI–28
GSTs (global symbol tables)
analyzing, *DCL A-M*, DCLI–23
deleting symbols from, *DCL A-M*, DCLI–153
entering symbol in, *DCL A-M*, DCLI–282

H

Header allocation
on disk volumes, *DCL A-M*, DCLI–260
HELP command, *DCL A-M*, DCLI–239 to DCLI–246
Help display
of default libraries, *DCL A-M*, DCLI–240
Help files, *DCL A-M*, DCLI–239
Help libraries
creating, *DCL A-M*, DCLI–239
user, *DCL A-M*, DCLI–243
HELP/MESSAGE command, *DCL A-M*, DCLI–247 to DCLI–252
Hexadecimal dump, *DCL A-M*, DCLI–192
Hibernation
See RUN command

I

IF command, *DCL A-M*, DCLI–253 to DCLI–255
Image files, *DCL A-M*, DCLI–22 to DCLI–24
See also PATCH command
analyzing, *DCL A-M*, DCLI–22
analyzing fixup section, *DCL A-M*, DCLI–23
analyzing global symbol table, *DCL A-M*, DCLI–23
analyzing patch text records, *DCL A-M*, DCLI–24
error analyzing of, *DCL A-M*, DCLI–22
invoking, *DCL N-Z*, DCLII–12
Images
continuing execution of, *DCL A-M*, DCLI–64
executing in detached process, *DCL N-Z*, DCLII–58
executing in subprocess, *DCL N-Z*, DCLII–58
hibernation
See RUN command
resuming execution of, *DCL A-M*, DCLI–64
running, *DCL N-Z*, DCLII–56

Images (cont'd)
specifying size with the RUN command, *DCL N-Z*, DCLII–62
terminating with EXIT command, *DCL A-M*, DCLI–230
wakeup
See RUN command
Indexes
creating source file with DSR, *DCL N-Z*, DCLII–69
Index files
placing on disk, *DCL A-M*, DCLI–261
INITIALIZE command, *DCL A-M*, DCLI–256 to DCLI–265
INITIALIZE/QUEUE command, *DCL A-M*, DCLI–266 to DCLI–280
Initializing
queues, *DCL A-M*, DCLI–266
tapes
using REPLY/BLANK_TAPE, *DCL N-Z*, DCLII–44
using REPLY/INITIALIZE_TAPE, *DCL N-Z*, DCLII–44
volumes, *DCL A-M*, DCLI–256
Input converter
DDIF, *DCL A-M*, DCLI–67
DTIF, *DCL A-M*, DCLI–68
Text, *DCL A-M*, DCLI–69
Input data stream
marking beginning of, *DCL A-M*, DCLI–119
marking end of, *DCL A-M*, DCLI–214
Input streams
defining for created process, *DCL N-Z*, DCLII–59
switching control to other processes, *DCL A-M*, DCLI–50
INQUIRE command, *DCL A-M*, DCLI–281 to DCLI–283
INSTALL command, *DCL A-M*, DCLI–284
Interactive help, *DCL A-M*, DCLI–242
ISO 9660 standard
DUMP utility, *DCL A-M*, DCLI–190

J

Job batch cards
end of, *DCL A-M*, DCLI–216
JOB card password, *DCL N-Z*, DCLII–10
JOB command, *DCL A-M*, DCLI–285 to DCLI–290
Job logical name tables
canceling entries, *DCL A-M*, DCLI–112
including logical name, *DCL A-M*, DCLI–43, DCLI–124
Job retention
determining, *DCL N-Z*, DCLII–22, DCLII–128, DCLII–431
Jobs
defining CPU time limit, *DCL A-M*, DCLI–271; *DCL N-Z*, DCLII–193, DCLII–388
deleting from queue, *DCL A-M*, DCLI–143, DCLI–150
redirecting to another queue, *DCL A-M*, DCLI–47
removing from queue
with ASSIGN/MERGE command, *DCL A-M*, DCLI–47

K

Kept Debugger, *DCL A-M*, DCLI–116
Keypad applications
setting for DECterm, *DCL A-M*, DCLI–105

L

Labels
command interpreter rules for, *DCL A-M*, DCLI–53, DCLI–235, DCLI–237
in command procedures, *DCL A-M*, DCLI–53, DCLI–235, DCLI–237
specifying for volumes, *DCL N-Z*, DCLII–242
volume header, *DCL A-M*, DCLI–256
writing on volume, *DCL A-M*, DCLI–256
Lexical functions, *DCL A-M*, DCLI–293
overview, *DCL A-M*, DCLI–291
Libraries
object module, *DCL A-M*, DCLI–28
LIBRARY command, *DCL A-M*, DCLI–416
LICENSE command, *DCL A-M*, DCLI–417
Licenses
displaying active, *DCL N-Z*, DCLII–293
LINK command, *DCL A-M*, DCLI–418
Link option specification records
analyzing in object files, *DCL A-M*, DCLI–28
Listing files in directory, *DCL A-M*, DCLI–170
Local symbols, *DCL A-M*, DCLI–1, DCLI–5
Local symbol tables
deleting symbols from, *DCL A-M*, DCLI–153
entering symbol in, *DCL A-M*, DCLI–282
Lock limits
specifying for detached processes, *DCL N-Z*, DCLII–62
specifying for subprocesses, *DCL N-Z*, DCLII–62
Logging in, *DCL A-M*, DCLI–419 to DCLI–422
Logging out, *DCL A-M*, DCLI–423
and device accesses, *DCL A-M*, DCLI–15
Logical name inclusion
in group logical name tables, *DCL A-M*, DCLI–42, DCLI–123
in job logical name tables, *DCL A-M*, DCLI–43, DCLI–124
in process logical name tables, *DCL A-M*, DCLI–43, DCLI–124

Logical name inclusion (cont'd)
in system logical name tables, *DCL A-M*, DCLI–43, DCLI–124
Logical names
assigning, *DCL A-M*, DCLI–41, DCLI–122
assigning to devices, *DCL A-M*, DCLI–15
canceling, *DCL A-M*, DCLI–111
creating, *DCL A-M*, DCLI–41, DCLI–122
creating a table, *DCL A-M*, DCLI–100
deassigning using CLOSE command, *DCL A-M*, DCLI–59
displaying
equivalence name for, *DCL N-Z*, DCLII–301, DCLII–366
translation of, *DCL N-Z*, DCLII–301, DCLII–366
process-permanent, *DCL N-Z*, DCLII–58
Logical name tables
canceling entries, *DCL A-M*, DCLI–113
canceling system entries, *DCL A-M*, DCLI–113
creating, *DCL A-M*, DCLI–100
deleting, *DCL A-M*, DCLI–111
displaying, *DCL N-Z*, DCLII–301
including logical names, *DCL A-M*, DCLI–43, DCLI–124
including system logical name, *DCL A-M*, DCLI–43, DCLI–124
Logical queues, *DCL A-M*, DCLI–268
deassigning, *DCL A-M*, DCLI–115
Login command procedures
executing, *DCL A-M*, DCLI–419
specifying alternate, *DCL A-M*, DCLI–419
LOGINOUT.EXE file
and detached processes, *DCL N-Z*, DCLII–61
LOGOUT command, *DCL A-M*, DCLI–423
message, *DCL A-M*, DCLI–423
multiple, *DCL A-M*, DCLI–423
Longword dump, *DCL A-M*, DCLI–193

M

MACRO–32 compiler, *DCL A-M*, DCLI–424
MACRO command, *DCL A-M*, DCLI–424 to DCLI–430
Mailboxes
process termination, *DCL N-Z*, DCLII–63
MAIL command, *DCL A-M*, DCLI–431
Mail utility (MAIL), *DCL A-M*, DCLI–431
Managers, queue
See Queue managers
Match size
specifying with DIFFERENCES command, *DCL A-M*, DCLI–164
/MEDIA_FORMAT qualifier
for INITIALIZE command, *DCL A-M*, DCLI–262; *DCL N-Z*, DCLII–165
Memory
displaying availability and use of
process balance slots, *DCL N-Z*, DCLII–307
process entry slots, *DCL N-Z*, DCLII–307
displaying error count for, *DCL N-Z*, DCLII–285
examining contents, *DCL A-M*, DCLI–217
modifying, *DCL A-M*, DCLI–155
replacing contents, *DCL A-M*, DCLI–155
MERGE command, *DCL A-M*, DCLI–432
Merging
of DIFFERENCES, *DCL A-M*, DCLI–165
queues, *DCL A-M*, DCLI–47
MESSAGE command, *DCL A-M*, DCLI–433
Message files
setting format, *DCL N-Z*, DCLII–167
Message logging
CONVERT/DOCUMENT command, *DCL A-M*, DCLI–70
Messages
online documentation, *DCL A-M*, DCLI–247
sending to terminals, *DCL N-Z*, DCLII–43
Modes of transferring files, *DCL A-M*, DCLI–223
Module header records
analyzing in object files, *DCL A-M*, DCLI–28
Module objects
analyzing, *DCL A-M*, DCLI–26
analyzing end-of-file records, *DCL A-M*, DCLI–27
Module traceback records
analyzing in object files, *DCL A-M*, DCLI–28
MONITOR command, *DCL A-M*, DCLI–434
MOUNT command, *DCL A-M*, DCLI–435
See also DEASSIGN command
See also DISMOUNT command
Movefile
controlling whether movefile operations are enabled, *DCL N-Z*, DCLII–136
determining if movefile operations are enabled, *DCL A-M*, DCLI–327
Multiprocessing systems
showing attached processor state, *DCL N-Z*, DCLII–259
starting attached processor, *DCL N-Z*, DCLII–382
stopping attached processors, *DCL N-Z*, DCLII–405

N

Names
See also Logical names
detached processes, *DCL N-Z*, DCLII–60
generic devices, *DCL A-M*, DCLI–15
logical
canceling, *DCL A-M*, DCLI–111
deassigning, *DCL A-M*, DCLI–59

Names (cont'd)
subprocesses, *DCL N-Z*, DCLII–60
symbol definition, *DCL A-M*, DCLI–1, DCLI–5
National character set (NCS) utility, *DCL N-Z*, DCLII–1
NCS command, *DCL N-Z*, DCLII–1
NCS utility
See National character set (NCS) utility
Network nodes
See also SET HOST command
See also SET HOST/DUP command
See also SET HOST/HSC command
and batch jobs, *DCL N-Z*, DCLII–430
connecting to a HSC, *DCL N-Z*, DCLII–152
connecting to a remote processor, *DCL N-Z*, DCLII–140
connecting to a storage controller, *DCL N-Z*, DCLII–150
Networks
pausing, *DCL N-Z*, DCLII–407
releasing from paused states, *DCL N-Z*, DCLII–407
restarting, *DCL N-Z*, DCLII–384
See also Networks, releasing from paused state
See also Networks, starting
starting, *DCL N-Z*, DCLII–384
stopping, *DCL N-Z*, DCLII–384
Node names
displaying, *DCL N-Z*, DCLII–368
Non OpenVMS systems
specifying remote files on, *DCL A-M*, DCLI–91
Nonpaged dynamic memory
displaying availability and use of, *DCL N-Z*, DCLII–307

O

Object files
analyzing, *DCL A-M*, DCLI–26
identifying
errors, *DCL A-M*, DCLI–26
Object module libraries, *DCL A-M*, DCLI–28
Object modules
analyzing, *DCL A-M*, DCLI–26
end-of-file records, *DCL A-M*, DCLI–27
Octal dump, *DCL A-M*, DCLI–193
ON command, *DCL N-Z*, DCLII–2 to DCLII–5
Online help, *DCL A-M*, DCLI–239
OPCOM (operator communication manager)
enabling terminal to receive messages from, *DCL N-Z*, DCLII–45
messages to users from, *DCL N-Z*, DCLII–52
OPEN command, *DCL N-Z*, DCLII–6 to DCLII–9
Open files, *DCL N-Z*, DCLII–6
displaying names, *DCL N-Z*, DCLII–267
Opening of queue and journal files, *DCL N-Z*, DCLII–397
OpenVMS RMS
See RMS
Operator communication manager
See OPCOM
Operators (computer)
See also REQUEST command
disabling status, *DCL N-Z*, DCLII–45
enabling status, *DCL N-Z*, DCLII–45
log file closing, *DCL N-Z*, DCLII–46
log file opening, *DCL N-Z*, DCLII–46
requesting replies from, *DCL N-Z*, DCLII–52
sending requests to, *DCL N-Z*, DCLII–52
Options files
CONVERT/DOCUMENT command, *DCL A-M*, DCLI–71
creating, *DCL A-M*, DCLI–71
Output converter
analysis, *DCL A-M*, DCLI–67
DDIF, *DCL A-M*, DCLI–68
DTIF, *DCL A-M*, DCLI–69
PostScript, *DCL A-M*, DCLI–70
Text, *DCL A-M*, DCLI–70
Output queues
See Queues
Output streams
defining for created process, *DCL N-Z*, DCLII–59
Overlaying files using the COPY command, *DCL A-M*, DCLI–84
Overriding
default command interpreter, *DCL A-M*, DCLI–419
magnetic tape overwrite protection, *DCL A-M*, DCLI–262
owner identification field, *DCL A-M*, DCLI–262
Owner identifier field
writing characters to, *DCL A-M*, DCLI–261
Ownership
specifying for volumes, *DCL A-M*, DCLI–263

P

/PAGE=SAVE qualifier navigation keys, *DCL A-M*, DCLI–166, DCLI–175, DCLI–193, DCLI–241; *DCL N-Z*, DCLII–74, DCLII–254, DCLII–268, DCLII–274, DCLII–282, DCLII–286, DCLII–294, DCLII–303, DCLII–309, DCLII–324, DCLII–334, DCLII–356, DCLII–369, DCLII–442
Paged dynamic memory
displaying availability and use of, *DCL N-Z*, DCLII–307
Parameters
passing to batch jobs, *DCL N-Z*, DCLII–429

Parameters (cont'd)
passing to command procedures, *DCL A-M*, DCLI–9, DCLI–53
specifying for command procedures, *DCL A-M*, DCLI–9
PASSWORD command, *DCL N-Z*, DCLII–10 to DCLII–11
Passwords
changing, *DCL N-Z*, DCLII–174
setting, *DCL N-Z*, DCLII–174
setting at login, *DCL A-M*, DCLI–419
PATCH command, *DCL N-Z*, DCLII–12
Patch text records
analyzing, *DCL A-M*, DCLI–24
Patch utility (PATCH)
changing code in, *DCL N-Z*, DCLII–12
invoking, *DCL N-Z*, DCLII–12
Pausing networks, *DCL N-Z*, DCLII–407
Pausing queues, *DCL N-Z*, DCLII–408
PHONE command, *DCL N-Z*, DCLII–13
Physical memory
displaying availability and use of, *DCL N-Z*, DCLII–307
Ports
displaying information, *DCL N-Z*, DCLII–368
PostScript output converter, *DCL A-M*, DCLI–70
processing options, *DCL A-M*, DCLI–73
PRINT command, *DCL N-Z*, DCLII–14 to DCLII–25
files, *DCL N-Z*, DCLII–14
procedure in batch job log, *DCL N-Z*, DCLII–238
Print queues
See Queues
Priorities
modifying processes, *DCL N-Z*, DCLII–182
specifying for batch jobs, *DCL N-Z*, DCLII–430
specifying for detached processes, *DCL N-Z*, DCLII–64
specifying for subprocesses, *DCL N-Z*, DCLII–64
Privileges
displaying processes, *DCL N-Z*, DCLII–325
modifying processes, *DCL N-Z*, DCLII–182
specifying for subprocesses, *DCL N-Z*, DCLII–64
Process dumps
analyzing of, *DCL A-M*, DCLI–30
Processes
See also Subprocesses
displaying
buffered I/O count, *DCL N-Z*, DCLII–351
characteristics of, *DCL N-Z*, DCLII–322
CPU time used, *DCL N-Z*, DCLII–351
current physical memory occupied, *DCL N-Z*, DCLII–351
current working set size, *DCL N-Z*, DCLII–351
Processes
displaying (cont'd)
identification, *DCL N-Z*, DCLII–324
information on, *DCL N-Z*, DCLII–247
names, *DCL N-Z*, DCLII–368
open file count, *DCL N-Z*, DCLII–351
page faults, *DCL N-Z*, DCLII–351
privilege, *DCL N-Z*, DCLII–325
status, *DCL N-Z*, DCLII–351
updated information about, *DCL N-Z*, DCLII–323, DCLII–324
displaying list of systems processes, *DCL N-Z*, DCLII–354
hibernation
with ATTACH command, *DCL A-M*, DCLI–50
image wakeup, *DCL N-Z*, DCLII–60
modifying characteristics of, *DCL N-Z*, DCLII–182
modifying working set default size, *DCL N-Z*, DCLII–245
placing in wait state, *DCL N-Z*, DCLII–447
priority
for detached processes, *DCL N-Z*, DCLII–64
for subprocesses, *DCL N-Z*, DCLII–64
privileges
specifying for subprocesses, *DCL N-Z*, DCLII–64
setting default devices or directories, *DCL N-Z*, DCLII–108
status
displaying current, *DCL N-Z*, DCLII–351
swap mode
enabling or disabling, *DCL N-Z*, DCLII–186
swapping
for created processes, *DCL N-Z*, DCLII–66
switching control of input stream, *DCL A-M*, DCLI–50
synchronizing with batch job, *DCL N-Z*, DCLII–436
working set
displaying quota and limit, *DCL N-Z*, DCLII–373
Processing options
analysis output converter, *DCL A-M*, DCLI–71
domain converter, *DCL A-M*, DCLI–75
in an options file, *DCL A-M*, DCLI–71
PostScript output converter, *DCL A-M*, DCLI–73
text output converter, *DCL A-M*, DCLI–72
Processors
showing state of attached, *DCL N-Z*, DCLII–259
showing state of secondary, *DCL N-Z*, DCLII–259
starting attached, *DCL N-Z*, DCLII–382

Processors (cont'd)
starting secondary, *DCL N-Z*, DCLII–382
stopping attached, *DCL N-Z*, DCLII–405
stopping secondary, *DCL N-Z*, DCLII–405
PRODUCT command, *DCL N-Z*, DCLII–26
Programs
continuing execution of, *DCL A-M*, DCLI–64
marking beginning of input stream, *DCL A-M*, DCLI–119
marking end of input stream, *DCL A-M*, DCLI–214
resuming execution of, *DCL A-M*, DCLI–64
Prompts
displaying in command procedures, *DCL A-M*, DCLI–281
Protection
default at disk initialization, *DCL A-M*, DCLI–259
defining at directory creation, *DCL A-M*, DCLI–97
defining at file creation, *DCL A-M*, DCLI–94
disk volumes, *DCL A-M*, DCLI–263
displaying default, *DCL N-Z*, DCLII–330
displaying security profiles, *DCL N-Z*, DCLII–347
establishing default, *DCL N-Z*, DCLII–191
magnetic tape volumes, *DCL A-M*, DCLI–263
modifying security profiles, *DCL N-Z*, DCLII–209
PSWRAP command, *DCL N-Z*, DCLII–27
PURGE command, *DCL N-Z*, DCLII–28 to DCLII–31
Purging files, *DCL N-Z*, DCLII–28
See also Deleting

Q

Queue and journal files, *DCL N-Z*, DCLII–397
Queue database files
closing, *DCL N-Z*, DCLII–413
Queue managers
failover, *DCL N-Z*, DCLII–398
restarting, *DCL N-Z*, DCLII–398, DCLII–413
starting, *DCL N-Z*, DCLII–397
stopping, *DCL N-Z*, DCLII–413
Queue options
assigning, *DCL N-Z*, DCLII–385
changing, *DCL A-M*, DCLI–266; *DCL N-Z*, DCLII–385
Queues
See also Print queues
activating autostart, *DCL A-M*, DCLI–210; *DCL N-Z*, DCLII–385
assigning, *DCL A-M*, DCLI–266
assigning logical names to, *DCL A-M*, DCLI–48
assigning to devices, *DCL A-M*, DCLI–48
autostart, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–386
Queues (cont'd)
batch jobs
displaying entries, *DCL N-Z*, DCLII–280, DCLII–331
entering command procedures in, *DCL N-Z*, DCLII–424
changing entries for batch and printer jobs, *DCL N-Z*, DCLII–122
changing options on, *DCL A-M*, DCLI–266; *DCL N-Z*, DCLII–385
creating, *DCL A-M*, DCLI–266
deactivating autostart, *DCL N-Z*, DCLII–414, DCLII–418
deassigning, *DCL A-M*, DCLI–115
defining CPU time limit for batch jobs, *DCL A-M*, DCLI–271; *DCL N-Z*, DCLII–193, DCLII–388
defining default working sets for batch jobs, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–197, DCLII–394, DCLII–432
defining working set extent for batch jobs, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–394, DCLII–433
defining working set quotas for batch jobs, *DCL A-M*, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–394, DCLII–433
deleting batch or print, *DCL A-M*, DCLI–150
deleting entries, *DCL A-M*, DCLI–143
designating as autostart, *DCL A-M*, DCLI–268
disabling autostart, *DCL A-M*, DCLI–182; *DCL N-Z*, DCLII–420
displaying default characteristics of system print queues, *DCL N-Z*, DCLII–320
displaying device entries, *DCL N-Z*, DCLII–280, DCLII–331
displaying entries, *DCL N-Z*, DCLII–331
ensuring availability of, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–386
establishing a print queue as spooled, *DCL N-Z*, DCLII–110
establishing base priority for batch jobs, *DCL N-Z*, DCLII–192
establishing base priority for jobs, *DCL A-M*, DCLI–269
execution of, *DCL A-M*, DCLI–267
designating autostart or nonautostart, *DCL A-M*, DCLI–268
failover of, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–386
generic, *DCL A-M*, DCLI–267
initializing, *DCL A-M*, DCLI–266
logical, *DCL A-M*, DCLI–268
merging jobs, *DCL A-M*, DCLI–47
modifying characteristics of a print queue, *DCL N-Z*, DCLII–179
pausing, *DCL N-Z*, DCLII–408
preparing for a shutdown, *DCL N-Z*, DCLII–421

Queues (cont'd)
preventing failover, *DCL A-M*, DCLI–182
reinitializing existing, *DCL A-M*, DCLI–266
releasing from paused states, *DCL N-Z*, DCLII–408
removing jobs from, *DCL A-M*, DCLI–47
restarting, *DCL N-Z*, DCLII–385, DCLII–414, DCLII–418
See also Queues, releasing from paused state
See also Queues, starting
restarting after deactivating for autostart, *DCL N-Z*, DCLII–414, DCLII–418
server, *DCL A-M*, DCLI–268
setting up for failover, *DCL A-M*, DCLI–269; *DCL N-Z*, DCLII–386
starting, *DCL N-Z*, DCLII–385
starting autostart, *DCL A-M*, DCLI–210; *DCL N-Z*, DCLII–385
starting queue managers, *DCL N-Z*, DCLII–397
stopping, *DCL N-Z*, DCLII–385, DCLII–414, DCLII–418
all, *DCL N-Z*, DCLII–420
before shutdown, *DCL A-M*, DCLI–182; *DCL N-Z*, DCLII–420
queue managers, *DCL N-Z*, DCLII–413
symbionts, *DCL A-M*, DCLI–268
types of, *DCL A-M*, DCLI–267
Quotas
AST limit, *DCL N-Z*, DCLII–61
batch jobs
working set, *DCL N-Z*, DCLII–433
CPU
for created processes, *DCL N-Z*, DCLII–66
of subprocesses process can create, *DCL N-Z*, DCLII–65
working set
for batch jobs, *DCL N-Z*, DCLII–433
modifying, *DCL N-Z*, DCLII–245
Quotation marks (‘‘ ’’)
in remote file specifications, *DCL A-M*, DCLI–91

R

Read check
with APPEND command, *DCL A-M*, DCLI–39
with COPY command, *DCL A-M*, DCLI–84
with INITIALIZE command, *DCL A-M*, DCLI–258
READ command, *DCL N-Z*, DCLII–32 to DCLII–35
Ready access
for directories on disk, *DCL A-M*, DCLI–257
RECALL command, *DCL N-Z*, DCLII–36 to DCLII–38
Record Management Services
See RMS
Records
analyzing
debugger information, *DCL A-M*, DCLI–27
global symbol directory, *DCL A-M*, DCLI–28
link option specification, *DCL A-M*, DCLI–28
module header, *DCL A-M*, DCLI–28
module traceback, *DCL A-M*, DCLI–28
patch text, *DCL A-M*, DCLI–24
relocation, *DCL A-M*, DCLI–29
analyzing end-of-file, *DCL A-M*, DCLI–27
comparing, *DCL A-M*, DCLI–160
reading, *DCL N-Z*, DCLII–32
writing to a file, *DCL N-Z*, DCLII–449
Recovering
EDT editing session, *DCL A-M*, DCLI–202
Releasing networks from paused states, *DCL N-Z*, DCLII–407
Releasing queues from paused states, *DCL N-Z*, DCLII–408
Relocation records
analyzing in object files, *DCL A-M*, DCLI–29
Remote files
specifying, *DCL A-M*, DCLI–91
RENAME command, *DCL N-Z*, DCLII–39 to DCLII–42
REPLY command, *DCL N-Z*, DCLII–43 to DCLII–51
See also INITIALIZE command
See also MOUNT command
See also REQUEST command
disabling operator status, *DCL N-Z*, DCLII–45
enabling operator status, *DCL N-Z*, DCLII–45
responding to file system requests, *DCL N-Z*, DCLII–44
responding to user requests, *DCL N-Z*, DCLII–44
REQUEST command, *DCL N-Z*, DCLII–52 to DCLII–53
Restarting
deactivated autostart queues, *DCL N-Z*, DCLII–414, DCLII–418
networks, *DCL N-Z*, DCLII–384
See also Starting networks
queues, *DCL N-Z*, DCLII–385
See also Starting queues
stopped queues, *DCL N-Z*, DCLII–414, DCLII–418
the queue manager, *DCL N-Z*, DCLII–398, DCLII–413

Restarting a network
See Releasing networks from paused states
Restarting a queue
See Releasing queues from paused states
Resuming execution
of command procedure, *DCL A-M*, DCLI–64
of DCL commands, *DCL A-M*, DCLI–64
of program, *DCL A-M*, DCLI–64
RETURN command, *DCL N-Z*, DCLII–54 to DCLII–55
Return key
to log in, *DCL A-M*, DCLI–419
Rights lists
modifying, *DCL N-Z*, DCLII–202
RMS
displaying default block count, *DCL N-Z*, DCLII–346
modifying defaults for, *DCL N-Z*, DCLII–205
Runaway tapes
stopping, *DCL A-M*, DCLI–257
RUN command
See also ATTACH command
See also SPAWN command
with images, *DCL N-Z*, DCLII–56, DCLII–57
abbreviating, *DCL N-Z*, DCLII–56
with processes, *DCL N-Z*, DCLII–58 to DCLII–68
creating detached processes, *DCL N-Z*, DCLII–66
Runoff
See DSR

S

Screen-oriented editors
EDT, *DCL A-M*, DCLI–200
EVE, *DCL A-M*, DCLI–209
SEARCH command, *DCL N-Z*, DCLII–70 to DCLII–77
Search lists, *DCL A-M*, DCLI–41, DCLI–122
Security
object protection, *DCL N-Z*, DCLII–209, DCLII–347
Security auditing, *DCL N-Z*, DCLII–84
Server queues, *DCL A-M*, DCLI–268
SET ACCOUNTING command, *DCL N-Z*, DCLII–81 to DCLII–83
SET AUDIT command, *DCL N-Z*, DCLII–84 to DCLII–96
SET BROADCAST command, *DCL N-Z*, DCLII–97 to DCLII–98
SET CARD_READER command, *DCL N-Z*, DCLII–99
SET CLUSTER/EXPECTED_VOTES command, *DCL N-Z*, DCLII–100 to DCLII–101
SET command, *DCL N-Z*, DCLII–78 to DCLII–80
SET COMMAND command, *DCL N-Z*, DCLII–102 to DCLII–104
SET CONTROL command, *DCL N-Z*, DCLII–105 to DCLII–106
SET DAY command, *DCL N-Z*, DCLII–107
SET DEFAULT command, *DCL N-Z*, DCLII–108 to DCLII–109
SET DEVICE command, *DCL N-Z*, DCLII–110 to DCLII–111
SET DEVICE/SERVED command, *DCL N-Z*, DCLII–112
SET DIRECTORY command, *DCL N-Z*, DCLII–113 to DCLII–115
SET DISPLAY command, *DCL N-Z*, DCLII–116 to DCLII–121
SET ENTRY command, *DCL N-Z*, DCLII–122 to DCLII–131
SET FILE command, *DCL N-Z*, DCLII–132 to DCLII–139
SET HOST command, *DCL N-Z*, DCLII–140 to DCLII–142
See also Network nodes
SET HOST/DTE command, *DCL N-Z*, DCLII–143
SET HOST/DUP command, *DCL N-Z*, DCLII–150, DCLII–151
See also Network nodes
SET HOST/HSC command, *DCL N-Z*, DCLII–152, DCLII–153
See also Network nodes
SET HOST/LAT command, *DCL N-Z*, DCLII–154 to DCLII–158
/FRAME qualifier, *DCL N-Z*, DCLII–156
SET HOST/RLOGIN command, *DCL N-Z*, DCLII–159 to DCLII–160
SET HOST/TELNET command, *DCL N-Z*, DCLII–161
SET HOST/TN3270 command, *DCL N-Z*, DCLII–162
SET KEY command, *DCL N-Z*, DCLII–163
SET LOGINS command, *DCL N-Z*, DCLII–164
SET MAGTAPE command, *DCL N-Z*, DCLII–165, DCLII–166
SET MAGTAPE/RETENSION command, *DCL N-Z*, DCLII–165
SET MESSAGE command, *DCL N-Z*, DCLII–167, DCLII–168
SET NETWORK command, *DCL N-Z*, DCLII–169
SET ON command, *DCL N-Z*, DCLII–172
SET OUTPUT_RATE command, *DCL N-Z*, DCLII–173
SET PASSWORD command, *DCL N-Z*, DCLII–174 to DCLII–176
SET PREFIX command, *DCL N-Z*, DCLII–177, DCLII–178

SET PRINTER command, *DCL N-Z*, DCLII–179 to DCLII–181
SET PROCESS command, *DCL N-Z*, DCLII–182 to DCLII–189
SET PROMPT command, *DCL N-Z*, DCLII–190
SET PROTECTION/DEFAULT command, *DCL N-Z*, DCLII–191
SET QUEUE command, *DCL N-Z*, DCLII–171, DCLII–192 to DCLII–199
SET RESTART_VALUE command, *DCL N-Z*, DCLII–200 to DCLII–201
SET RIGHTS_LIST command, *DCL N-Z*, DCLII–202 to DCLII–204
  attributes, *DCL N-Z*, DCLII–202
    Dynamic, *DCL N-Z*, DCLII–202
    No Access, *DCL N-Z*, DCLII–202
    Resource, *DCL N-Z*, DCLII–202
    Subsystem, *DCL N-Z*, DCLII–202
SET RMS_DEFAULT command, *DCL N-Z*, DCLII–205 to DCLII–208
SET SECURITY command, *DCL N-Z*, DCLII–209 to DCLII–216
SET SYMBOL command, *DCL N-Z*, DCLII–219 to DCLII–221
SET TERMINAL command, *DCL N-Z*, DCLII–222 to DCLII–235
SET TIME command, *DCL N-Z*, DCLII–236 to DCLII–237
SET VERIFY command, *DCL N-Z*, DCLII–238 to DCLII–240
SET VOLUME command, *DCL N-Z*, DCLII–241 to DCLII–244
SET WORKING_SET command, *DCL N-Z*, DCLII–245, DCLII–246
$SEVERITY global symbol, *DCL N-Z*, DCLII–172
  changing, *DCL A-M*, DCLI–230; *DCL N-Z*, DCLII–54
Shareable images
  analyzing, *DCL A-M*, DCLI–22
  file analyzing, *DCL A-M*, DCLI–22
  patching, *DCL N-Z*, DCLII–12
Shareable volumes
  dismounting, *DCL A-M*, DCLI–186
  initializing disk as, *DCL A-M*, DCLI–264
Shared device
  dismounting, *DCL A-M*, DCLI–187
Shelving
  See also Automatic unshelving
  controlling whether file is shelvable, *DCL N-Z*, DCLII–137
  determining if file is shelvable, *DCL A-M*, DCLI–176, DCLI–328
  determining if file is shelved, *DCL A-M*, DCLI–176, DCLI–177, DCLI–328
SHOW ACCOUNTING command, *DCL N-Z*, DCLII–250, DCLII–251
SHOW AUDIT command, *DCL N-Z*, DCLII–252 to DCLII–255
SHOW BROADCAST command, *DCL N-Z*, DCLII–256, DCLII–257
SHOW CLUSTER command, *DCL N-Z*, DCLII–258
SHOW command, *DCL N-Z*, DCLII–247 to DCLII–249
  summary of options, *DCL N-Z*, DCLII–247
SHOW CPU command, *DCL N-Z*, DCLII–259 to DCLII–263
SHOW DEFAULT command, *DCL N-Z*, DCLII–264, DCLII–265
SHOW DEVICES command, *DCL N-Z*, DCLII–266 to DCLII–272
SHOW DEVICES/SERVED command, *DCL N-Z*, DCLII–273 to DCLII–277
SHOW DISPLAY command, *DCL N-Z*, DCLII–278, DCLII–279
SHOW ENTRY command, *DCL N-Z*, DCLII–280 to DCLII–284
SHOW ERROR command, *DCL N-Z*, DCLII–285, DCLII–287
SHOW INTRUSION command, *DCL N-Z*, DCLII–288 to DCLII–290
SHOW KEY command, *DCL N-Z*, DCLII–291, DCLII–292
SHOW LICENSE command, *DCL N-Z*, DCLII–293 to DCLII–300
SHOW LOGICAL command, *DCL N-Z*, DCLII–301 to DCLII–306
SHOW MEMORY command, *DCL N-Z*, DCLII–307 to DCLII–316
SHOW NETWORK command, *DCL N-Z*, DCLII–317 to DCLII–319
SHOW PRINTER command, *DCL N-Z*, DCLII–320, DCLII–321
SHOW PROCESS command, *DCL N-Z*, DCLII–322 to DCLII–329
SHOW PROTECTION command, *DCL N-Z*, DCLII–330
SHOW QUEUE/CHARACTERISTICS command, *DCL N-Z*, DCLII–338, DCLII–339
SHOW QUEUE command, *DCL N-Z*, DCLII–331 to DCLII–337
SHOW QUEUE/FORM command, *DCL N-Z*, DCLII–340, DCLII–341
SHOW QUEUE/MANAGERS command, *DCL N-Z*, DCLII–342 to DCLII–344
SHOW QUOTA command, *DCL N-Z*, DCLII–345
SHOW RMS_DEFAULT command, *DCL N-Z*, DCLII–346
SHOW SECURITY command, *DCL N-Z*, DCLII–347 to DCLII–350

SHOW STATUS command, *DCL N-Z*, DCLII–351
SHOW SYMBOL command, *DCL N-Z*, DCLII–352, DCLII–353
SHOW SYSTEM command, *DCL N-Z*, DCLII–354 to DCLII–361
SHOW TERMINAL command, *DCL N-Z*, DCLII–362, DCLII–364
See also SET TERMINAL command; SHOW TERMINAL command
SHOW TIME command, *DCL N-Z*, DCLII–365
SHOW TRANSLATION command, *DCL N-Z*, DCLII–366, DCLII–367
SHOW USERS command, *DCL N-Z*, DCLII–368 to DCLII–372
SHOW WORKING_SET command, *DCL N-Z*, DCLII–373 to DCLII–374
SHOW ZONE command, *DCL N-Z*, DCLII–375
Shutdown
preparing a queue for, *DCL N-Z*, DCLII–421
stopping autostart queues before, *DCL A-M*, DCLI–182
stopping queues before, *DCL N-Z*, DCLII–420
SORT command, *DCL N-Z*, DCLII–376
SPAWN command, *DCL N-Z*, DCLII–377 to DCLII–381
See also ATTACH command, *DCL A-M*, DCLI–50
START/CPU command, *DCL N-Z*, DCLII–382 to DCLII–383
Starting networks, *DCL N-Z*, DCLII–384
See also Restarting networks
Starting queues, *DCL N-Z*, DCLII–385
See also Restarting queues
autostart, *DCL A-M*, DCLI–210, DCLI–269; *DCL N-Z*, DCLII–385
nonautostart, *DCL A-M*, DCLI–269
queue manager, *DCL N-Z*, DCLII–397
START/NETWORK command, *DCL N-Z*, DCLII–384
START/QUEUE command, *DCL N-Z*, DCLII–385 to DCLII–396
START/QUEUE/MANAGER command, *DCL N-Z*, DCLII–397 to DCLII–400
START/ZONE command, *DCL N-Z*, DCLII–401
Status code
controlling command interpreter response to, *DCL N-Z*, DCLII–172
$STATUS global symbol, *DCL N-Z*, DCLII–172
changing, *DCL A-M*, DCLI–230; *DCL N-Z*, DCLII–54
STOP command, *DCL N-Z*, DCLII–402 to DCLII–404
See also Ctrl/C key sequence
See also Ctrl/Y key sequence
See also EXIT command
runaway tapes, *DCL A-M*, DCLI–257
STOP/CPU command, *DCL N-Z*, DCLII–405, DCLII–406
STOP/NETWORK command, *DCL N-Z*, DCLII–407
Stopping networks
See also Pausing networks
Stopping queues, *DCL N-Z*, DCLII–414, DCLII–420
See also Pausing queues
abruptly, *DCL N-Z*, DCLII–418
Stopping the queue managers, *DCL N-Z*, DCLII–413
STOP/QUEUE/ABORT command, *DCL N-Z*, DCLII–409, DCLII–410
STOP/QUEUE command, *DCL N-Z*, DCLII–408
STOP/QUEUE/ENTRY command, *DCL N-Z*, DCLII–411, DCLII–412
STOP/QUEUE/MANAGER/CLUSTER command, *DCL N-Z*, DCLII–413
STOP/QUEUE/NEXT command, *DCL N-Z*, DCLII–414
See also DELETE/QUEUE command
STOP/QUEUE/REQUEUE command, *DCL N-Z*, DCLII–415 to DCLII–417
STOP/QUEUE/RESET command, *DCL N-Z*, DCLII–418, DCLII–419
STOP/QUEUES/ON_NODE command, *DCL N-Z*, DCLII–420 to DCLII–422
STOP/ZONE command, *DCL N-Z*, DCLII–423
:= (String assignment) command, *DCL A-M*, DCLI–5 to DCLI–8
Structure levels
defining for disks, *DCL A-M*, DCLI–264
Subdirectories
creating, *DCL A-M*, DCLI–96
SUBMIT command, *DCL A-M*, DCLI–288; *DCL N-Z*, DCLII–424 to DCLII–434
Subprocesses
See also SPAWN command
accounting, *DCL N-Z*, DCLII–61
assigning resource quotas to, *DCL N-Z*, DCLII–59
creating with RUN command, *DCL N-Z*, DCLII–58
creating with SPAWN command, *DCL N-Z*, DCLII–377
defining attributes, *DCL N-Z*, DCLII–59
defining equivalence names for process-permanent logical names, *DCL N-Z*, DCLII–58
displaying characteristics of, *DCL N-Z*, DCLII–322
displaying quotas, *DCL N-Z*, DCLII–325
image hibernation, *DCL N-Z*, DCLII–60
naming with RUN/PROCESS_NAME, *DCL N-Z*, DCLII–60
scheduling wakeup, *DCL N-Z*, DCLII–63

Subprocesses (cont'd)
specifying default working set, *DCL N-Z*, DCLII–66
specifying quotas, *DCL N-Z*, DCLII–62
switching control of input stream, *DCL A-M*, DCLI–50
SUBROUTINE command, *DCL A-M*, DCLI–54, DCLI–55; *DCL N-Z*, DCLII–435
Subroutines
termination of GOSUB, *DCL N-Z*, DCLII–54
SUMSLP editor, *DCL A-M*, DCLI–205
Swapping
for created processes, *DCL N-Z*, DCLII–66
processes
enabling or disabling swap mode, *DCL N-Z*, DCLII–186
Symbionts
customer-written, *DCL A-M*, DCLI–268
default, *DCL A-M*, DCLI–268
special purpose, *DCL A-M*, DCLI–268
Symbolic names
defining, *DCL A-M*, DCLI–1, DCLI–5
Symbols
assigning value with READ command, *DCL N-Z*, DCLII–32
binary overlay in, *DCL A-M*, DCLI–1
character overlays in, *DCL A-M*, DCLI–6
deleting
from global symbol tables, *DCL A-M*, DCLI–153
from local symbol tables, *DCL A-M*, DCLI–153
displaying, *DCL N-Z*, DCLII–352
general assignment, *DCL A-M*, DCLI–1
interactive assignment in command procedures, *DCL A-M*, DCLI–281
masking, *DCL N-Z*, DCLII–219
string assignment, *DCL A-M*, DCLI–5
SYNCHRONIZE command, *DCL N-Z*, DCLII–436 to DCLII–437
SYS$ERROR logical name
specifying equivalence name with RUN command, *DCL N-Z*, DCLII–62
SYS$INPUT logical name
specifying equivalence name with RUN command, *DCL N-Z*, DCLII–63
SYS$LOST directory, *DCL N-Z*, DCLII–136
SYS$OUTPUT
displaying file on, *DCL N-Z*, DCLII–438
SYS$OUTPUT logical name
specifying equivalence name with RUN command, *DCL N-Z*, DCLII–64
SYS$SYLOGIN logical name
executing, *DCL A-M*, DCLI–419
System login images
and detached processes, *DCL N-Z*, DCLII–61
System messages
online documentation, *DCL A-M*, DCLI–247
System performance
displaying availability and use of resources, *DCL N-Z*, DCLII–307
Systems
accessing, *DCL A-M*, DCLI–419
changing
date, *DCL N-Z*, DCLII–236
passwords, *DCL N-Z*, DCLII–174
time, *DCL N-Z*, DCLII–236
displaying
information on, *DCL N-Z*, DCLII–247
status, *DCL N-Z*, DCLII–247
System time
changing, *DCL N-Z*, DCLII–236

T

TA90E tape drive
support for, *DCL N-Z*, DCLII–165
using /MEDIA_FORMAT qualifier, *DCL N-Z*, DCLII–165
Tapes
disabling operator status, *DCL N-Z*, DCLII–45
dismounting, *DCL A-M*, DCLI–186
enabling operator status, *DCL N-Z*, DCLII–45
establishing operational status for, *DCL N-Z*, DCLII–110
initializing using REPLY command, *DCL N-Z*, DCLII–44
modifying RMS defaults for file operations, *DCL N-Z*, DCLII–205
TECO editor, *DCL A-M*, DCLI–206
Terminal emulators
creating, *DCL A-M*, DCLI–104
Terminals
See also SET TERMINAL command; SHOW TERMINAL command
default characteristics, *DCL A-M*, DCLI–419
displaying characteristics of, *DCL N-Z*, DCLII–362
displaying files at, *DCL N-Z*, DCLII–438
establishing as spooled, *DCL N-Z*, DCLII–110
modifying characteristics of, *DCL N-Z*, DCLII–222
sending messages to, *DCL N-Z*, DCLII–43
virtual, *DCL A-M*, DCLI–61, DCLI–184
Terminal sessions
logging in, *DCL A-M*, DCLI–419
logging out, *DCL A-M*, DCLI–423
Terminations
of command procedure, *DCL A-M*, DCLI–230
of GOSUB subroutine, *DCL N-Z*, DCLII–54
of terminal session, *DCL A-M*, DCLI–423
Testing
the value of an expression, *DCL A-M*, DCLI–253

Text
analyzing in object files, *DCL A-M*, DCLI–29
Text editors
See EDT editor
See EVE editor
See SUMSLP editor
See TECO editor
Text file formatting
See DSR
Text input converter, *DCL A-M*, DCLI–69
Text output converter, *DCL A-M*, DCLI–70
processing options, *DCL A-M*, DCLI–72
THEN keyword
and IF command, *DCL A-M*, DCLI–253
Time
changing system, *DCL N-Z*, DCLII–236
CPU quota for created processes, *DCL N-Z*, DCLII–66
CPU used by current process, *DCL N-Z*, DCLII–351
displaying, *DCL N-Z*, DCLII–365
Timed retention
specifying, *DCL N-Z*, DCLII–23, DCLII–129, DCLII–432
TPU
See DECTPU
Transfer modes
EXCHANGE/NETWORK command, *DCL A-M*, DCLI–223
True expression
and IF command, *DCL A-M*, DCLI–253
TYPE command, *DCL N-Z*, DCLII–438 to DCLII–444
TZK10 tape cartridge drives, *DCL N-Z*, DCLII–165

U

UAFs (user authorization files)
and detached processes, *DCL N-Z*, DCLII–61
UICs (user identification codes)
specifying for directories, *DCL A-M*, DCLI–97
specifying for files, *DCL A-M*, DCLI–94
Unloading devices
with DISMOUNT command, *DCL A-M*, DCLI–188
UNLOCK command, *DCL N-Z*, DCLII–445
Unlocking files, *DCL N-Z*, DCLII–445
User authorization files
See UAFs
User identification codes
See UICs
User libraries
See Help libraries
User names
specifying at login, *DCL A-M*, DCLI–419
Users
displaying disk quota of, *DCL N-Z*, DCLII–345
displaying interactive terminal name of, *DCL N-Z*, DCLII–368
displaying names on system, *DCL N-Z*, DCLII–368
displaying process identification code (PID) of, *DCL N-Z*, DCLII–368
recording name on disk volume, *DCL N-Z*, DCLII–243
responding to requests, *DCL N-Z*, DCLII–44
User-specified job retention
PRINT/RETAIN command, *DCL N-Z*, DCLII–21
SET ENTRY/RETAIN command, *DCL N-Z*, DCLII–128
SUBMIT/RETAIN command, *DCL N-Z*, DCLII–430

V

Value test in expression, *DCL A-M*, DCLI–253
Verification
modifying for command procedures, *DCL N-Z*, DCLII–238
Version limits
for files in directory, *DCL A-M*, DCLI–97
Version numbers
assigning to files, *DCL A-M*, DCLI–224
VIEW command, *DCL N-Z*, DCLII–446
Virtual devices, *DCL A-M*, DCLI–220, DCLI–342
Virtual memory
examining contents, *DCL A-M*, DCLI–217
replacing contents, *DCL A-M*, DCLI–155
Virtual terminals
connecting, *DCL A-M*, DCLI–61
disconnecting from, *DCL A-M*, DCLI–184
VMScluster envirnments
specifying cluster size on disk, *DCL A-M*, DCLI–258
VMScluster environments
dismounting volumes on, *DCL A-M*, DCLI–187
Volume accessibility fields
writing characters to, *DCL A-M*, DCLI–261
Volumes
deleting disk files, *DCL A-M*, DCLI–138
dismounting disk and tape, *DCL A-M*, DCLI–186
displaying disk quota, *DCL N-Z*, DCLII–345
dumping, *DCL A-M*, DCLI–190
Files–11
modifying characteristics of, *DCL N-Z*, DCLII–241
recording name on, *DCL N-Z*, DCLII–243
initializing, *DCL A-M*, DCLI–256
label, *DCL A-M*, DCLI–256
protecting, *DCL A-M*, DCLI–263

Volumes (cont'd)
specifying maximum file number, *DCL A-M*, DCLI–262
specifying ownership, *DCL A-M*, DCLI–263
Volume sets
dismounting, *DCL A-M*, DCLI–188

W

WAIT command, *DCL N-Z*, DCLII–447 to DCLII–448
Wait state
delaying command processing, *DCL N-Z*, DCLII–447
inducing to synchronize process with batch job, *DCL N-Z*, DCLII–436
placing current process in, *DCL N-Z*, DCLII–447
Wakeup requests
canceling, *DCL A-M*, DCLI–57
canceling request, *DCL N-Z*, DCLII–60
scheduling with RUN command, *DCL N-Z*, DCLII–60
Windows
displaying size and count for open files, *DCL N-Z*, DCLII–270
Word dumps, *DCL A-M*, DCLI–195
Working sets
batch jobs
defining default for, *DCL A-M*, DCLI–278; *DCL N-Z*, DCLII–394, DCLII–432
defining extent for, *DCL A-M*, DCLI–278, DCLI–288; *DCL N-Z*, DCLII–394, DCLII–433
defining quotas for batch job, *DCL N-Z*, DCLII–394
defining quotas for batch jobs, *DCL A-M*, DCLI–288; *DCL N-Z*, DCLII–198, DCLII–433
displaying
limit for process, *DCL N-Z*, DCLII–373
displaying quotas, *DCL N-Z*, DCLII–373
modifying default size, *DCL N-Z*, DCLII–245
specifying default
for detached processes, *DCL N-Z*, DCLII–66
for subprocesses, *DCL N-Z*, DCLII–66
specifying quotas, *DCL N-Z*, DCLII–63
Write check
with APPEND command, *DCL A-M*, DCLI–39
with COPY command, *DCL A-M*, DCLI–85
with INITIALIZE command, *DCL A-M*, DCLI–258
WRITE command, *DCL N-Z*, DCLII–449 to DCLII–451
Writing records to a file, *DCL N-Z*, DCLII–449