VMS Software

VSI OpenVMS

VSI DECset for OpenVMS Language-Sensitive Editor/Source Code Analyzer Reference Manual

Operating System and Version: VSI OpenVMS x86-64 Version 9.2-2 or higher
VSI OpenVMS IA-64 Version 8.4-1H1 or higher
VSI OpenVMS Alpha Version 8.4-2L1 or higher

Software Version: DECset Version 12.7

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

VSI DECset for OpenVMS Language-Sensitive Editor/Source Code Analyzer Reference Manual

Legal Notice

Table of Contents

Preface xi
1. About VSI xi
2. Intended Audience xi
3. Document Structure xi
4. Related Documents xi
5. References to Other Products xii
6. OpenVMS Documentation xii
7. VSI Encourages Your Comments xii
8. Conventions xii

Chapter 1. Command Dictionary 1
1.1. Executing Commands 1
1.2. Canceling Commands 2
1.3. Commands in Buffers 2
1.4. Command Categories 2
1.4.1. Editing Session Control Commands 2
1.4.2. Text Manipulation Commands 2
1.4.3. Entering Source Code Commands 3
1.4.4. SCA Navigation Commands 3
1.4.5. SCA Library Commands 3
1.4.6. SCA Query Commands 4
1.4.7. Query Session Manipulation Commands 4
1.4.8. Commands for Compiling Source Code and Reviewing Errors 4
1.4.9. Indenting Source Code Commands 4
1.4.10. Cursor Movement Commands 4
1.4.11. Screen Manipulation Commands 5
1.4.12. File and Buffer Manipulation Commands 5
1.4.13. Program Design Commands 5
1.4.14. Commands for Tailoring the Environment 6
1.4.15. Help and Status Commands 6
1.4.16. CMS Commands 7

Chapter 2. Command Descriptions 9
@ (file-specification) 9
ALIGN 10
ANALYZE 11
ATTACH 13
BOX COPY 14
BOX CUT 14
BOX DRAW 15
BOX PASTE 15
BOX LOWERCASE 16
BOX UPPERCASE 17
CALL 17
CANCEL MARK 19
CANCEL SELECT_MARK 20
CAPITALIZE WORD 20
CENTER LINE 21
CHANGE CASE 21
CHANGE DIRECTION 22
CHANGE INDENTATION 22

CHANGE TEXT_ENTRY_MODE 24
CHANGE WINDOW_MODE 25
CHECK LANGUAGE 25
CLOSE BUFFER 27
CMS 28
COLLAPSE 29
COMPILE 30
CONTINUE 32
CONVERT LIBRARY 33
CREATE LIBRARY 34
CUT 35
DCL 38
DEFINE ADJUSTMENT 39
DEFINE ALIAS 43
DEFINE COMMAND 44
DEFINE KEY 45
DEFINE KEYWORDS 50
DEFINE LANGUAGE 51
DEFINE PACKAGE 58
DEFINE PARAMETER 60
DEFINE PLACEHOLDER 62
DEFINE ROUTINE 67
DEFINE TAG 70
DEFINE TOKEN 71
DELETE ADJUSTMENT 75
DELETE ALIAS 76
DELETE BUFFER 77
DELETE COMMAND 77
DELETE KEY 78
DELETE KEYWORDS 79
DELETE LANGUAGE 79
DELETE LIBRARY 80
DELETE MODULE 81
DELETE OVERVIEW 82
DELETE PACKAGE 83
DELETE PARAMETER 84
DELETE PLACEHOLDER 84
DELETE QUERY 85
DELETE ROUTINE 86
DELETE TAG 87
DELETE TOKEN 88
DELETE WINDOW 89
DO 89
END DEFINE 91
END REVIEW 92
ENLARGE WINDOW 92
ENTER COMMENT 93
ENTER LINE 95
ENTER PSEUDOCODE 97
ENTER SPACE 98
ENTER SPECIAL 99
ENTER TAB 99

ENTER TEXT 100
ERASE CHARACTER 100
ERASE LINE 102
ERASE PLACEHOLDER 104
ERASE SELECTION 106
ERASE WORD 106
EXIT 108
EXPAND 109
EXTEND 112
EXTRACT ADJUSTMENT 113
EXTRACT ALIAS 114
EXTRACT KEYWORDS 115
EXTRACT LANGUAGE 116
EXTRACT MODULE 117
EXTRACT PACKAGE 118
EXTRACT PARAMETER 119
EXTRACT PLACEHOLDER 120
EXTRACT ROUTINE 122
EXTRACT TAG 123
EXTRACT TOKEN 124
FILL 125
FIND 127
FOCUS 131
GOTO BOTTOM 132
GOTO BUFFER 132
GOTO CHARACTER 134
GOTO COMMAND 135
GOTO DECLARATION 136
GOTO FILE 138
GOTO LINE 140
GOTO MARK 143
GOTO PAGE 143
GOTO PLACEHOLDER 144
GOTO QUERY 146
GOTO REVIEW 146
GOTO SCREEN 147
GOTO SOURCE 148
GOTO TOP 150
GOTO WORD 151
HELP 152
IMPORT 154
INCLUDE 155
INSPECT 156
LINE 159
LOAD 160
LOWERCASE WORD 161
MODIFY LANGUAGE 162
NEXT BUFFER 168
NEXT ERROR 169
NEXT OCCURRENCE 170
NEXT QUERY 170
NEXT STEP 171

NEXT SYMBOL 171
NEXT WINDOW 172
ONE WINDOW 173
OTHER WINDOW 173
PASTE 174
PREVIOUS BUFFER 175
PREVIOUS ERROR 176
PREVIOUS OCCURRENCE 176
PREVIOUS QUERY 177
PREVIOUS STEP 177
PREVIOUS SYMBOL 178
PREVIOUS WINDOW 178
QUIT 179
QUOTE 180
READ 181
RECALL 182
RECOVER BUFFER 182
REDO 183
REFRESH 184
REORGANIZE 184
REPEAT 185
REPLACE 186
REPORT 187
RESERVE 190
REVIEW 191
SAVE ENVIRONMENT 192
SAVE QUERY 193
SAVE SECTION 195
SEARCH 196
SELECT ALL 200
SET AUTO_ERASE 200
SET CMS 201
SET CURSOR 203
SET DEFAULT_DIRECTORY 204
SET DIRECTORY 205
SET FONT 205
SET FORWARD 206
SET INDENTATION 207
SET INSERT 208
SET JOURNALING 209
SET LANGUAGE 210
SET LEFT_MARGIN 211
SET LIBRARY 212
SET MARK 213
SET MAX_UNDO 214
SET MODE 215
SET MODIFY 217
SET NOAUTO_ERASE 218
SET NOJOURNALING 218
SET NOLANGUAGE 219
SET NOLIBRARY 220
SET NOMODIFY 221

SET NOOUTPUT_FILE 222
SET NOOVERVIEW 223
SET NOSOURCE_DIRECTORY 224
SET NOWRAP 224
SET OUTPUT_FILE 225
SET OVERSTRIKE 226
SET OVERVIEW 227
SET READ_ONLY 228
SET REVERSE 229
SET RIGHT_MARGIN 229
SET SCREEN 230
SET SCROLL_MARGINS 232
SET SEARCH 233
SET SELECT_MARK 234
SET SOURCE_DIRECTORY 235
SET TAB_INCREMENT 236
SET WRAP 237
SET WRITE 238
SHIFT 239
SHOW ADJUSTMENT 239
SHOW ALIAS 240
SHOW BUFFER 241
SHOW CMS 243
SHOW COMMAND 244
SHOW DEFAULT_DIRECTORY 244
SHOW DIRECTORY 245
SHOW KEY 245
SHOW KEYWORDS 247
SHOW LANGUAGE 248
SHOW LIBRARY 249
SHOW MARK 250
SHOW MAX_UNDO 251
SHOW MODE 251
SHOW MODULE 252
SHOW PACKAGE 254
SHOW PARAMETER 255
SHOW PLACEHOLDER 256
SHOW QUERY 257
SHOW ROUTINE 258
SHOW SCREEN 259
SHOW SEARCH 260
SHOW SOURCE_DIRECTORY 260
SHOW SUMMARY 261
SHOW TAG 261
SHOW TOKEN 262
SHOW VERSION 263
SHRINK WINDOW 264
SPAWN 265
SPELL 265
SPLIT WINDOW 266
SUBSTITUTE 267
TAB 270

TOGGLE SELECT_MARK 270
TWO WINDOWS 271
UNDO 272
UNDO ENTER COMMENT 273
UNERASE 273
UNEXPAND 275
UNRESERVE 275
UNTAB 276
UPPERCASE WORD 276
VERIFY 277
VIEW SOURCE 278
WHAT LINE 279
WRITE 280

Appendix A. Interfacing to DECTPU Procedures 283

A.1. DECTPU Variables and Procedures 283
A.2. Guidelines for User-Written TPU Procedures 286
A.2.1. Adding User-Written TPU Procedures 286
A.2.2. DECTPU Programming with Hidden Records in LSE 287
A.3. Supplemental DECTPU Built-Ins 288
A.3.1. LSE$DO_COMMAND (String) 288
A.3.2. LSE$GET_ENVIRONMENT(String, Keyword) 288
A.3.3. GET_INFO (buffer, ˝language˝) 288
A.3.4. GET_INFO (buffer, ˝overviews˝) 289
A.3.5. GET_INFO(COMMAND_LINE, item) 289
A.3.6. LSE$FIND_OPEN_COMMENT (marker) 289
A.3.7. LSE$FIND_CLOSE_COMMENT (marker) 289
A.3.8. LSE$IS_OVERVIEW [(marker)] 290
A.3.9. LSE$IS_VISIBLE [(marker)] 290
A.3.10. LSE$MOVE_HORIZONTAL (integer) 290
A.3.11. LSE$MOVE_VERTICAL (integer) 290
A.3.12. LSE$MOVE_BY_SOURCE (integer) 290
A.3.13. LSE$MAKE_VISIBLE (marker |range) 290
A.3.14. LSE$NEAREST_VISIBLE (marker) 290
A.3.15. LSE$SOURCE_ONLY (range) 290
A.3.16. LSE$MOVE_TEXT and LSE$COPY_TEXT (string |range |buffer) 291
A.3.17. SET (LSE$LANGUAGE, buffer, language) 291
A.3.18. SET (LSE$OVERVIEWS, buffer, on/off) 291
A.3.19. TPU Built-ins for the SCA Callable Interface 292

Appendix B. Language-Specific Information 293

B.1. VSI Fortran 293
B.1.1. Token and Placeholder Definitions 293
B.1.2. Entering and Erasing Text 294
B.1.3. Indentation 294
B.2. VSI COBOL 294

Appendix C. Packages 295

C.1. DECTPU Procedures for the Package Facility 295
C.2. Example Procedures 297

Appendix D. LSE and EVE Commands 307

Appendix E. Portable and VMSLSE Commands 311

Appendix F. Providing 7-Bit Terminal Support for Code Elision 329

Appendix G. TPU Pattern Style 331
G.1. User Interface 331
G.2. Partial Pattern Assignment Variables 332
G.3. New Line 332
G.4. Errors 333
G.5. Global Variables 333
G.6. Pattern Variables 333
G.7. Use for Developing DTM User Filters 334

Preface

This manual contains reference material on the VSI Language-Sensitive Editor for OpenVMS (SCA) and the VSI Source Code Analyzer for OpenVMS (SCA). The LSE commands are in the VMSLSE format, and the SCA commands are in the VMS format. See the *VSI DECset for OpenVMS Language-Sensitive Editor Command-Line Interface and Callable Routines Reference Manual* for information on the VMSLSE, VMS, and Portable command language syntaxes.

1. About VSI

VMS Software, Inc. (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

2. Intended Audience

This manual is for experienced programmers, technical writers, and technical managers.

3. Document Structure

This manual has a command dictionary and appendices that contain reference information. The structure is as follows:

- The Command Dictionary contains an alphabetical list of all the LSE and SCA commands that are available from command-line mode.
- *Appendix A, "Interfacing to DECTPU Procedures"* provides information on writing your own DECTPU procedures.
- *Appendix B, "Language-Specific Information"* contains information of interest to VSI Fortran and VSI COBOL programmers.
- *Appendix C, "Packages"* describes how to write your own DECTPU routines for use with the package facility.
- *Appendix D, "LSE and EVE Commands"* contains a list of the EVE commands with the corresponding LSE commands.
- *Appendix E, "Portable and VMSLSE Commands"* contains a list of the Portable commands with their equivalent VMSLSE commands.
- *Appendix F, "Providing 7-Bit Terminal Support for Code Elision"* contains information about using the OpenVMS Terminal Fallback Facility to translate double-angle brackets to single-angle brackets on 7-bit terminals.

4. Related Documents

The following documents might also be helpful when using LSE and SCA:

- See your installation guide for installation instructions for LSE and SCA.
- The *VSI DECset for OpenVMS Guide to Language-Sensitive Editor* contains tutorial information on using the VSI DECset for OpenVMS Language-Sensitive Editor.

- The *VSI DECset for OpenVMS Guide to VSI Source Code Analyzer* contains tutorial information on using the VSI DECset for OpenVMS Source Code Analyzer.
- The *VSI DECset for OpenVMS Language-Sensitive Editor Command-Line Interface and Callable Routines Reference Manual* contains command-line interface and callable routine information for the VSI DECset for OpenVMS Language-Sensitive Editor.
- The *VSI DECset for OpenVMS Source Code Analyzer Command-Line Interface and Callable Routines Reference Manual* contains callable routine and query information for the VSI DECset for OpenVMS Source Code Analyzer.
- The *DEC Text Processing Utility Reference Manual* describes the VSI Text Processing Utility features, including the high-level procedural language available for use with LSE.
- *Using VSI DECset for OpenVMS Systems* describes how to use the DECset products with other OpenVMS software development facilities to create an effective development environment.

5. References to Other Products

Some older products that DECset components previously worked with might no longer be available or supported by VSI. Any reference in this manual to such products does not imply actual support, or that recent interoperability testing has been conducted with these products.

Note

These references serve only to provide examples to those who continue to use these products with DECset.

Refer to the Software Product Description for a current list of the products that the DECset components are warranted to interact with and support.

6. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com.

7. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: <docinfo@vmssoftware.com>. Users who have VSI OpenVMS support contracts through VSI can contact <support@vmssoftware.com> for help with this product.

8. Conventions

VMScluster systems are now referred to as OpenVMS Cluster systems. Unless otherwise specified, references to OpenVMS Cluster systems or clusters in this document are synonymous with VMScluster systems.

The contents of the display examples for some utility 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 OpenVMS Alpha and Integrity servers, that behavior is described in text and rendered, as appropriate, in separate examples.

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:

Convention	Meaning
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 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
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.)
…	A horizontal ellipsis in examples indicates 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.
. . .	A vertical ellipsis indicates 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 you must enclose the options in parentheses if you choose more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
[|]	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are options; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold text	This typeface represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER= *name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
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.

Convention	Meaning
	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 at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

Chapter 1. Command Dictionary

This chapter describes all the commands for VSI Language-Sensitive Editor (LSE) for OpenVMS, and Source Code Analyzer (SCA) for OpenVMS. Section 1.1 describes how to enter command mode from your editing session. Section 1.2 describes how to cancel commands and return to your editing session. Section 1.3 describes how to execute commands typed into a buffer. Section 1.4 groups LSE and SCA commands by function. The remainder of the dictionary contains the individual command descriptions.

Note that if a section, such as Qualifiers or Parameters, is not applicable to a specific command, the section does not appear under that command.

1.1. Executing Commands

As described in the related user guides, all the LSE and SCA commands described in this manual can be entered at the LSE command line. Invoke LSE (in either character-cell or DECwindows format) with the LSEDIT command at the OpenVMS prompt. You have the option of executing a single command and returning to keypad mode, or executing several commands without leaving command mode, as follows:

- *To execute only one command*—Enter command mode by pressing the Do key, or use the PF1 and COMMAND (KP7) sequence. Type the command at the `LSE Command>` prompt and press the Return key. The command executes and LSE returns to keypad mode. In DECwindows only, you can also enter command mode by clicking MB1 below the status line.

- *To execute several commands*—Enter command mode by pressing Ctrl/Z. Type the first command at the `LSE>` prompt and terminate the command string by pressing the Return key. You will still be at the `LSE>` prompt after the command executes. Press Ctrl/Z or enter the CONTINUE command to return to keypad mode.

LSE provides multiple command recall; by using the up and down arrow keys at the `LSE>` or `LSECommand>` prompt, you can recall any of the commands you entered during your current editing session.

LSE provides two command languages: VMSLSE (the commands described in this manual) and Portable. See the *VSI DECset for OpenVMS Language-Sensitive Editor Command-Line Interface and Callable Routines Reference Manual* for information on setting your default command language and bypassing the default with individual commands.

Appendix E, "Portable and VMSLSE Commands" contains a translation table that lists VMSLSE equivalents to Portable commands.

Descriptions of Portable commands are available only in online Help. To get help on Portable commands, execute one of the following commands:

- VMSLSE to Portable translation table

  ```
  LSE>  PLSE HELP VMSLSE_Command_translation_Table
  ```

- Top-level Help for Portable commands

  ```
  LSE>  PLSE HELP
  ```

- List of Portable commands with the same first word

  ```
  LSE>  PLSE HELP SET
  ```

This example would generate a window containing a list similar to the following:

```
SET ADJUSTMENT COMPRESS
SET ADJUSTMENT COUNT
SET ADJUSTMENT CURRENT
.
.
.
```

Get Help by moving the cursor to the desired command and pressing the Return key.

1.2. Canceling Commands

To cancel a command, press Ctrl/Z in response to a prompt. For example, pressing Ctrl/Z in response to the Search for: prompt cancels the SEARCH command. Pressing Ctrl/Z in response to the `LSE>` or `LSECommand>` prompt returns you to keypad editing.

Pressing Ctrl/C while the REPEAT or DO/BUFFER command is executing terminates that command.

1.3. Commands in Buffers

You can execute commands that have been typed into a buffer. At the LSE> prompt,enter the DO command with the /BUFFER qualifier and supply the name of the buffer containing the commands you want (see the individual command descriptions for more information).

1.4. Command Categories

The following lists identify the related LSE and SCA commands and tasks. For information on a command, see its individual description in the Command Descriptions section of this manual.

1.4.1. Editing Session Control Commands

ATTACH	REPEAT
CONTINUE	SET DEFAULT_DIRECTORY
DCL	SET FONT
DO	SET JOURNALING
EXIT	SET MAX_UNDO
GOTO COMMAND	SET NOJOURNALING
QUIT	SPAWN
RECALL	

1.4.2. Text Manipulation Commands

BOX COPY	FILL
BOX CUT	LOWERCASE WORD
BOX DRAW	PASTE
BOX PASTE	QUOTE
BOX LOWERCASE	REDO

BOX UPPERCASE	SELECT ALL
CANCEL SELECT_MARK	SET AUTO_ERASE
CAPITALIZE WORD	SET FORWARD
CENTER LINE	SET INSERT
CHANGE CASE	SET NOAUTO_ERASE
CHANGE DIRECTION	SET NOWRAP
CHANGE INDENTATION	SET OVERSTRIKE
CHANGE TEXT_ENTRY_MODE	SET REVERSE
CUT	SET SELECT_MARK
ENTER LINE	SET WRAP
ENTER SPACE	SPELL
ENTER SPECIAL	SUBSTITUTE
ENTER TAB	TAB
ENTER TEXT	TOGGLE SELECT_MARK
ERASE CHARACTER	UNERASE
ERASE LINE	UNEXPAND
ERASE PLACEHOLDER	UNTAB
ERASE SELECTION	UPPERCASE WORD
ERASE WORD	UNDO
EXPAND	

1.4.3. Entering Source Code Commands

ENTER COMMENT	SET LANGUAGE
ENTER PSEUDOCODE	SET NOAUTO_ERASE
ERASE PLACEHOLDER	SET NOLANGUAGE
EXPAND	UNDO ENTER COMMENT
GOTO PLACEHOLDER	UNERASE
SET AUTO_ERASE	UNEXPAND

1.4.4. SCA Navigation Commands

EXPAND	NEXT SYMBOL
GOTO SOURCE	PREVIOUS OCCURRENCE
IMPORT	PREVIOUS STEP
NEXT OCCURRENCE	PREVIOUS SYMBOL
NEXT STEP	UNEXPAND

1.4.5. SCA Library Commands

ANALYZE	REORGANIZE

CONVERT LIBRARY	SET LIBRARY
CREATE LIBRARY	SET NOLIBRARY
DELETE LIBRARY	SHOW LIBRARY
DELETE MODULE	SHOW MODULE
EXTRACT MODULE	VERIFY
LOAD	

1.4.6. SCA Query Commands

FIND	INSPECT
GOTO DECLARATION	

1.4.7. Query Session Manipulation Commands

DELETE QUERY	PREVIOUS QUERY
GOTO QUERY	SHOW QUERY
NEXT QUERY	

1.4.8. Commands for Compiling Source Code and Reviewing Errors

COMPILE	NEXT STEP
END REVIEW	PREVIOUS ERROR
GOTO REVIEW	PREVIOUS STEP
GOTO SOURCE	REVIEW
NEXT ERROR	

1.4.9. Indenting Source Code Commands

ALIGN	SET TAB_INCREMENT
CHANGE INDENTATION	SET WRAP
ENTER TAB	TAB
FILL	UNTAB
SET INDENTATION	

1.4.10. Cursor Movement Commands

CANCEL MARK	GOTO TOP
CHANGE DIRECTION	GOTO WORD
GOTO BOTTOM	LINE
GOTO MARK	SET FORWARD
GOTO PAGE	SET MARK

GOTO PLACEHOLDER	SET REVERSE
GOTO SCREEN	SET SEARCH
GOTO SOURCE	

1.4.11. Screen Manipulation Commands

CHANGE WINDOW_MODE	OTHER WINDOW
DELETE WINDOW	PREVIOUS WINDOW
ENLARGE WINDOW	REFRESH
GOTO BUFFER	SET SCREEN
GOTO FILE	SET SCROLL_MARGINS
GOTO SCREEN	SHIFT
GOTO SOURCE	SHRINK WINDOW
NEXT WINDOW	SPLIT WINDOW
ONE WINDOW	TWO WINDOWS

1.4.12. File and Buffer Manipulation Commands

CHANGE DIRECTION	SET INSERT
CHANGE TEXT_ENTRY_MODE	SET LEFT_MARGIN
CLOSE BUFFER	SET MODIFY
CUT	SET NOMODIFY
DELETE BUFFER	SET NOOUTPUT_FILE
GOTO BUFFER	SET NOSOURCE_DIRECTORY
GOTO FILE	SET OUTPUT_FILE
GOTO SOURCE	SET OVERSTRIKE
INCLUDE	SET READ_ONLY
NEXT BUFFER	SET REVERSE
PASTE	SET RIGHT_MARGIN
PREVIOUS BUFFER	SET SOURCE_DIRECTORY
READ	SET TAB_INCREMENT
RECOVER BUFFER	SET WRAP
SET DEFAULT_DIRECTORY	SET WRITE
SET DIRECTORY	SHOW BUFFER
SET FORWARD	WRITE
SET INDENTATION	

1.4.13. Program Design Commands

COLLAPSE	EXTRACT KEYWORDS
DEFINE ADJUSTMENT	EXTRACT TAG

DEFINE KEYWORDS	FOCUS
DEFINE TAG	REPORT
DELETE ADJUSTMENT	SET NOOVERVIEW
DELETE KEYWORDS	SET OVERVIEW
DELETE TAG	SHOW ADJUSTMENT
ENTER COMMENT	SHOW KEYWORDS
ENTER PSEUDOCODE	SHOW TAG
EXPAND	UNDO ENTER COMMENT
EXTRACT ADJUSTMENT	VIEW SOURCE

1.4.14. Commands for Tailoring the Environment

CALL	DELETE PLACEHOLDER
CHECK LANGUAGE	DELETE ROUTINE
DEFINE ADJUSTMENT	DELETE TAG
DEFINE ALIAS	DELETE TOKEN
DEFINE COMMAND	DO
DEFINE KEY	END DEFINE
DEFINE KEYWORDS	EXTEND
DEFINE LANGUAGE	EXTRACT ADJUSTMENT
DEFINE PACKAGE	EXTRACT ALIAS
DEFINE PARAMETER	EXTRACT KEYWORDS
DEFINE PLACEHOLDER	EXTRACT LANGUAGE
DEFINE ROUTINE	EXTRACT PACKAGE
DEFINE TAG	EXTRACT PARAMETER
DEFINE TOKEN	EXTRACT PLACEHOLDER
DELETE ADJUSTMENT	EXTRACT ROUTINE
DELETE ALIAS	EXTRACT TAG
DELETE COMMAND	EXTRACT TOKEN
DELETE KEY	MODIFY LANGUAGE
DELETE KEYWORDS	SAVE ENVIRONMENT
DELETE LANGUAGE	SAVE SECTION
DELETE PACKAGE	SET MODE
DELETE PARAMETER	SET SEARCH

1.4.15. Help and Status Commands

HELP	SHOW MODULE
SHOW ADJUSTMENT	SHOW PACKAGE
SHOW ALIAS	SHOW PARAMETER

SHOW BUFFER	SHOW PLACEHOLDER
SHOW CMS	SHOW QUERY
SHOW COMMAND	SHOW ROUTINE
SHOW DEFAULT_DIRECTORY	SHOW SCREEN
SHOW DIRECTORY	SHOW SEARCH
SHOW KEY	SHOW SOURCE_DIRECTORY
SHOW KEYWORDS	SHOW SUMMARY
SHOW LANGUAGE	SHOW TAG
SHOW LIBRARY	SHOW TOKEN
SHOW MARK	SHOW VERSION
SHOW MAX_UNDO	WHAT LINE
SHOW MODE	

1.4.16. CMS Commands

CMS	SET CMS
REPLACE	SHOW CMS
RESERVE	UNRESERVE

Chapter 2. Command Descriptions

This chapter describes the LSE and SCA commands in alphabetical order. To aid in differentiating these commands, the following notations appear under the command name:

Notation	Explanation
No notation	LSE standalone commands.
SCA Command	SCA standalone commands. These commands are valid any time you are using SCA, whether or not you are using LSE.
SCA Required	LSE commands that are valid only if you are using SCA with LSE.

In describing DECwindows menu equivalents for commands, the following terms are used:

Term	Description of action
Button	To activate, press MB1 on an item.
Pop-up menu	To activate, press MB2 on the first path item;follow the path while holding down MB2.
Pull-down menu	To activate, press MB1 on the first path item;follow the path while holding down MB1.

Note

LSE follows the quoting rules of the VSI Command Language (DCL). All references to quoted strings mean that LSE expects double quotation marks (").

In the command descriptions that follow,the defaults for qualifiers are indicated by (D).

@ (file-specification)

@ (file-specification) — Allows the execution of SCA commands contained in a specified file.

Format

`@ (file-specification)`

Description

The use of command files containing query definitions allows a common set of queries to be used interactively indifferent SCA sessions.

Related Commands

`SAVE_QUERY`

Example

The following queries could be used to describe all the names that might be associated with the Year 2000 problem:

```
$ SCA
SCA> @Y2000
SCA> FIND/OUT=Y2000.LIS @Y2000_QUERY AND OCC=REFERENCE
```

The command file is also usable in DECwindows mode, as follows:

1. Select Commands/Enter Commands . . .
2. Enter command: SET COMMAND LANGUAGE VMS.
3. Enter command: @y2000.
4. Select Cross Reference Query window.
5. Set the name field to "@y2000_query".
6. Set usage to Reference.
7. Issue the query.

Use the SCA/LSE interface to look at the references found.

ALIGN

ALIGN — Aligns comments within the current selected range without performing a fill operation.

Format

ALIGN

Qualifiers	Defaults
/COMMENT_COLUMN= CONTEXT_DEPENDENT	/COMMENT_COLUMN= CONTEXT_DEPENDENT
/COMMENT_COLUMN=number	/COMMENT_COLUMN= CONTEXT_DEPENDENT

Qualifier

/COMMENT_COLUMN=CONTEXT_DEPENDENT (D)
/COMMENT_COLUMN=number

The /COMMENT_COLUMN=CONTEXT_DEPENDENT qualifier specifies that the comment column should be determined from the context. LSE finds the first trailing comment in the range, uses the starting position of that comment as the comment column, and adjusts all subsequent comments to conform with the first. This is the default.

The /COMMENT_COLUMN=number qualifier specifies the column in which to align the comments. All trailing comments in the range are aligned with the specified column number, which must be an integer in the range 1 to 131.

Description

The ALIGN command aligns all trailing comments with a particular column. The column in which you position the comments can be either explicitly specified (using the /COMMENT_COLUMN=number qualifier) or based on context.

This command operates on each line in the range, in sequence. For each line, LSE checks to see whether the line has a trailing comment. If not, it proceeds to the next line.

If there is a trailing comment, LSE either inserts or deletes spaces or tabs as necessary to get the comment to align. If there is no room for the comment on the line (that is, if the noncommented text extends beyond the comment column), the comment is aligned one space after the end of the noncommented text.

DECwindows Interface Equivalent

Pull-down menu: Edit → Align

Related Commands

FILL

Example

The following is a sample of commented code:

1.
```
IF (col >= R_Margin) THEN     (* This is the start of a *)
        BEGIN                 (* bracketed comment sequence that *)
        VAR x: INTEGER;       (* extends over several lines *)
```

Entering the ALIGN command causes LSE to rearrange the text as follows:

2.
```
IF (col >= R_Margin) THEN (* This is the start of a          *)
        BEGIN             (* bracketed comment sequence that *)
        VAR x: INTEGER;   (* extends over several lines      *)
```

ANALYZE

ANALYZE — Creates an analysis data file that describes a source file.

Format

ANALYZE file-spec[, . . .]

Qualifiers	Defaults
/[NO]DESIGN[=design-option]	/NODESIGN
/LANGUAGE=language	

Qualifiers	Defaults
/[NO]LOG	/LOG
/OUTPUT[=file-spec]	/OUTPUT=file-name.ANA

Qualifiers

/DESIGN[=design-option]
/NODESIGN (D)

Indicates that the source file should be processed as a program design language. The design options are as follows:

Option	Description
COMMENTS	The ANALYZE command looks inside comments for design information. Information about comments is included in the analysis data file. Any errors detected are reported.
NOCOMMENTS	The ANALYZE command ignores comments.
PLACEHOLDERS	The ANALYZE command treats LSE placeholders as valid syntax. Placeholders are reported in the analysis datafile.
NOPLACEHOLDERS	The ANALYZE command does not report placeholders in the .ANA file. It does not report errors if placeholders are encountered.

If you specify the /DESIGN qualifier, the default is /DESIGN= (COMMENTS,PLACEHOLDERS). If you do not specify this qualifier, the default is /NODESIGN.

/LANGUAGE=language

Specifies the language of the source file. By default, the language is determined by the file type of the source file.

/LOG (D)
/NOLOG

Indicates whether each analyzed file is reported.

/OUTPUT[=file-spec]
/OUTPUT=file-name.ANA (D)

Specifies the analysis data file to be created. The default is /OUTPUT=*filename*.ANA, where *file-name*is the name of the first source file specified as the parameter to this command.

Parameter

`file-spec[, . . . ]`

Specifies the files to be analyzed. You can use wildcards with the *file-spec* parameter. Within LSE, the current buffer is analyzed by default.

Description

The ANALYZE command creates an analysis data file to describe a source file. The analysis data files produced by this command contain a minimal description of the source file. These files describe the source file primarily as a set of references to unbound names.

With the ANALYZE command, you can use SCA with languages not directly supported by SCA. Do not use this command with those languages that do support SCA. To identify those languages that support SCA, see the *VSI DECset for OpenVMS Source Code Analyzer Command-Line Interface and Callable Routines Reference Manual* or the DECset Software Product Description (SPD).

The ANALYZE command understands the language-specific rules for forming names (identifiers), comments, quoted strings, and placeholders. It assumes that tokens are reserved words, and does not include them in the analysis data file. It processes placeholders and comments depending on the setting of the /DESIGN qualifier.

You must have a language defined in an environment file to use the ANALYZE command with that language. Based on the description of the language in that file, this command analyzes the source file.

The ANALYZE command uses the LSE environment files to determine the appropriate language based on the file type, or uses the language specified with the /LANGUAGE qualifier. It uses the same logical names as LSE, (LSE$ENVIRONMENT and LSE$SYSTEM_ENVIRONMENT) to access the environment files.

For information about defining your own language, see the chapter on defining LSE templates in the *Guide to Language-Sensitive Editor for OpenVMS Systems*.

The REPORT command requires that LSE be installed even if you are using this command from the SCA command line.

Related

DEFINE LANGUAGE

LOAD

Examples

1. `LSE>` ***ANALYZE/LANGUAGE=EXAMPLE PROG1.EXAMPLE***

   Produces an analysis data file that describes an EXAMPLE language source file.

2. `LSE>` ***ANALYZE/DESIGN=(NOPLACEHOLDERS) PROG2.SDML***

   Produces an analysis data file and indicates that the source file should be processed as a program design language. Placeholders are not reported in the .ANA file. By default, information about comments are reported. The language is SDML, as determined by the file type of the source file.

ATTACH

ATTACH — Allows you to switch control of your terminal to another process. Note, that this function is not available in DECwindows; any attempt to invoke it creates an error.

Format

ATTACH [subprocess-name]

Parameter

subprocess-name

Specifies the name of the process to which you want to connect. If you do not specify a process name, LSE connects you to the parent process.

Description

The ATTACH command switches control of your terminal to another process,just as the DCL command ATTACH does at the dollar sign ($) prompt. To return to LSE from another process, use the DCL command ATTACH. Use the LOGOUT command to return to LSE only from a subprocess.

Related Commands

SPAWN

Example

```
LSE> ATTACH SMITH_1
```

Switches control to the process SMITH_1.

BOX COPY

BOX COPY — N/A

Format

BOX COPY

Description

The BOX COPY command copies the currently selected box to the default location (i.e. the DECwindows clipboard or the paste buffer).

Example

```
LSE> BOX COPY
```

BOX CUT

BOX CUT — N/A

Format

BOX CUT

Qualifiers

/PAD

Indicates that the area of the cut is to be padded with spaces.

Description

The BOX CUT command moves the currently selected text to the default location (the DECwindows clipboard or the paste buffer).

Example

```
LSE> BOX CUT
```

BOX DRAW

BOX DRAW — N/A

Format

BOX DRAW

Description

The BOX DRAW command draws a box in the overstrike mode. The box is drawn using the plus sign (+) for the corners, the vertical bar (|) for the sides and a hyphen (-) for the top and bottom.

Examples

A selection that includes all the upper case letters (of letter B) is made using the BOX DRAW command.

```
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeBBBBBBBBBBBBBBBBBIIIIIIIIII
IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
IIIIIIIIBBBBBBBBBBBBBBBBBeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

If the BOX DRAW command is issued for the preceding selection, the following is displayed:

```
LSE>  BOX DRAW
```

```
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeee+----------------+IIIIIIIIII
IIIIIIII|BBBBBBBBBBBBBBB|IIIIIIIIII
IIIIIIII|BBBBBBBBBBBBBBB|IIIIIIIIII
IIIIIIII+----------------+eeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

BOX PASTE

BOX PASTE — N/A

Format

BOX PASTE

Qualifiers

/OVERSTRIKE

Indicates that the paste is performed in the overstrike mode, the default selection is the insert mode.

Description

The BOX PASTE command copies the contents of the default location to a box with the top left hand corner at the current position.

Examples

A selection that includes all the upper case letters (of letter B) is made using the BOX PASTE command.

```
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeBBBBBBBBBBBBBBBBBIIIIIIIIII
IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
IIIIIIIIBBBBBBBBBBBBBBBBBeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

If the BOX PASTE command is issued for the preceding selection, the following is displayed:

```
LSE>  BOX PASTE

              BBBBBBBBBBBBBBBBB
              BBBBBBBBBBBBBBBBB
              BBBBBBBBBBBBBBBBB
              BBBBBBBBBBBBBBBBB
```

BOX LOWERCASE

BOX LOWERCASE — N/A

Format

BOX LOWERCASE

Description

The BOX LOWERCASE command changes the case of the text in the selected box to lowercase.

Examples

A selection that includes all the upper case letters (of letter B) is made using the BOX LOWERCASE command.

```
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

```
        eeeeeeeeBBBBBBBBBBBBBBBBBIIIIIIIIII
        IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
        IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
        IIIIIIIIBBBBBBBBBBBBBBBBBeeeeeeeeee
        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

If the BOX LOWERCASE command is issued for the preceding selection, the following is displayed:

```
LSE>  BOX LOWERCASE

        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
        eeeeeeeebbbbbbbbbbbbbbbbbIIIIIIIIII
        IIIIIIIIbbbbbbbbbbbbbbbbbIIIIIIIIII
        IIIIIIIIbbbbbbbbbbbbbbbbbIIIIIIIIII
        IIIIIIIIbbbbbbbbbbbbbbbbbeeeeeeeeee
        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

BOX UPPERCASE

BOX UPPERCASE — N/A

Format

BOX UPPERCASE

Description

The BOX UPPERCASE command changes the case of the text in the selected box to uppercase.

Examples

A selection that includes all the lower case letters (of letter b) is made using the BOX UPPERCASE command.

```
        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
        eeeeeeeebbbbbbbbbbbbbbbbbIIIIIIIIII
        IIIIIIIIbbbbbbbbbbbbbbbbbIIIIIIIIII
        IIIIIIIIbbbbbbbbbbbbbbbbbIIIIIIIIII
        IIIIIIIIbbbbbbbbbbbbbbbbbeeeeeeeeee
        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

If the BOX UPPERCASE command is issued for the preceding selection, the following is displayed:

```
LSE>  BOX UPPERCASE

        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
        eeeeeeeeBBBBBBBBBBBBBBBBBIIIIIIIIII
        IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
        IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
        IIIIIIIIBBBBBBBBBBBBBBBBBIIIIIIIIII
        eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

CALL

CALL — Calls the specified VSI Text Processing Utility (DECTPU) procedure.

Format

CALL DECTPU-procedure-name [additional-parameters]

Parameters

DECTPU-procedure-name

Indicates the name of the DECTPU procedure you want to call.

additional-parameters

Contains information to be passed to the procedure as a single string. The called procedure must then parse and interpret this string.

Description

The CALL command, in combination with the DEFINE COMMAND command, provides a means for defining new commands implemented in the DECTPU language. Because the additional parameters are passed to the called procedure without being parsed, these commands have a flexible syntax.

Related Commands

DEFINE COMMAND

DO/TPU

Example

The following DECTPU procedure issues a DIRECTORY command from within LSE:

```
PROCEDURE dir (dir_params)
    ! Description:
    !   Issues a DCL DIRECTORY command in a subprocess.  The output is
    !   written to the DIRECTORY buffer.  The DIRECTORY buffer is
    !   mapped to the current window.
    !
    ! Parameter:
    !   dir_params - a string beginning with "$".  The text following
    !       the "$" contains parameters and qualifiers to be passed to
    !       the DIRECTORY command.
    !       The "$" is used to provide a parameter for the call to
    !       this procedure when no parameters for the DIRECTORY
    !       command were specified.
    !         LOCAL dir_process, cmd;
    IF GET_INFO(dir_buffer, "TYPE") <> BUFFER THEN
        dir_buffer := CREATE_BUFFER("DIRECTORY");
        SET(NO_WRITE,dir_buffer);
    ENDIF;
    erase(dir_buffer);
    ! Build the DIRECTORY command, picking up parameters that were
    ! passed in.
    cmd := 'DIRECTORY '+SUBSTR(dir_params,2,LENGTH(dir_params)-1);
    ! Create a subprocess and execute the command.
```

```
        dir_process := CREATE_PROCESS (dir_buffer, cmd);
        lse$do_command('GOTO BUFFER DIRECTORY');
        DELETE (dir_process);     ENDPROCEDURE
```

To define this procedure, put it in a buffer and compile it using the DO/TPU command. To use the procedure, define a command named DIR, as follows:

```
LSE> DEFINE COMMAND DIR "CALL DIR $"
```

To get a directory listing, enter your newly defined DIR command, as follows:

```
LSE> DIR/SIZE/DATE
```

Note the use of the dollar sign ($) to cause the CALL command to always invoke the procedure named DIR with a parameter, even if you specify nothing else on the command line of the command DIR. The dollar sign also prevents qualifiers on the command DIR from being interpreted as an attempt to place qualifiers on the DECTPU *procedure-name* parameter named DIR.

CANCEL MARK

CANCEL MARK — Cancels the specified marker set by a SET MARK command.

Format

CANCEL MARK

Parameter

marker-name

Names the marker to be canceled; a wildcard marker name is allowed. If you do not specify a name for the marker, LSE cancels any marker at the current cursor position.

Description

The CANCEL MARK command causes LSE to remove the specified marker from the text and to delete the marker name.

DECwindows Interface Equivalent

Pull-down menu: Navigate > Cancel Mark

Related Commands

SET MARK

Example

```
LSE> CANCEL MARK 1
```

Deletes the marker named 1 from the inventory of markers in your current buffer.

CANCEL SELECT_MARK

CANCEL SELECT_MARK — Cancels the selected range of the SET SELECT_MARK command.

Format

CANCEL SELECT_MARK

Description

The CANCEL SELECT_MARK command cancels the effect of the SET SELECT_MARK command. If a block or range of text is displayed in reverse video, the CANCELSELECT_MARK command returns the text to its normal display.

Keypad Equivalent

Key	Keypad Mode
PF1-Keypad period (.) **RESET**	EDT LK201, EDT VT100, EVE LK201
PF1-E4 **SELECT**	EVE LK201

Related Commands

SET SELECT_MARK

CAPITALIZE WORD

CAPITALIZE WORD — Capitalizes the first letter of the current word, or words, in a selected range.

Format

CAPITALIZE WORD

Description

The CAPITALIZE WORD command capitalizes the first letter of the word following the cursor, or the word that the cursor is on. If a selected range is active,all the words within that range are capitalized.

If a word is already in uppercase letters, the command changes all but the first letter to lowercase. The cursor then moves to the first letter of the word following the target word or selected range.

DECwindows Interface Equivalent

Pull-down menu: Edit > Capitalize

Related Commands

CHANGE CASE

LOWERCASE WORD

UPPERCASE WORD

CENTER LINE

CENTER LINE — Centers the current line between the left and right margins.

Format

CENTER LINE

Description

The CENTER LINE command centers text on the line that the cursor is on. You can place the cursor anywhere on the line to be centered.

DECwindows Interface Equivalent

Pull-down menu: Edit > Center Line

CHANGE CASE

CHANGE CASE — Changes the case of a letter, or letters, in a selected range.

Format

CHANGE CASE

Description

The CHANGE CASE command changes the case of letters.

If you select a range of text by using the SET SELECT_MARK command,the case of each letter in the selected range changes.

Keypad Equivalent

Key	**Keypad Mode**
PF1-KP1 **CHNGCASE**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

Related Commands

CAPITALIZE WORD

LOWERCASE WORD

UPPERCASE WORD

CHANGE DIRECTION

CHANGE DIRECTION — Changes the current direction of the current buffer between forward and reverse.

Format

CHANGE DIRECTION

Description

The CHANGE DIRECTION command changes the current direction attribute of the current buffer. The buffer's status line indicates whether the current direction is forward or reverse. The direction affects the operation of such commands as GOTO, ERASE, SEARCH, SUBSTITUTE, CHANGE INDENTATION, and CHANGE CASE.

With the DECwindows interface, you can switch directions by moving the mouse cursor to Forward or Reverse on the status line, then pressing MB1.

Keypad Equivalent

Key	Keypad Mode
F11 **FORWARD REVERSE**	EVE LK201, EDT LK201
PF3 **FORWARD REVERSE**	EVE VT100
None	EDT VT100

DECwindows Interface Equivalent

Buffer status line $\begin{Bmatrix} \text{Forward} \\ \text{Reverse} \end{Bmatrix}$

Related Commands

SET FORWARD

SET REVERSE

CHANGE INDENTATION

CHANGE INDENTATION — Adds or removes leading blanks and tabs from lines.

Format

CHANGE INDENTATION

Qualifiers	Defaults
/CURRENT	/CURRENT

Qualifiers	Defaults
/FORWARD	/CURRENT
/[NO]HOLD	/HOLD
/REVERSE	/CURRENT

Qualifiers

/CURRENT (D)

Specifies the current direction for the change in indentation.

/FORWARD

Specifies the forward direction for the change in indentation.

/HOLD (D)
/NOHOLD

Specifies whether the selected range is canceled by this command. Use the /HOLD qualifier to keep the selected range active so you can repeat this command to make incremental changes in indentation.

/REVERSE

Specifies the reverse direction for the change in indentation.

Description

The CHANGE INDENTATION command adds or removes blanks and tabs from the line that the cursor is on, and sets the new indentation of the current line as the current indentation level.

If you select a range of text by using the SET SELECT_MARK command, the CHANGEINDENTATION command adds or removes blanks and tabs from each line of text in the selected range.

Keypad Equivalent

CHANGE INDENTATION/FORWARD

Key	Keypad Mode
PF1- > **IND FWD**	EDT LK201, EDT VT100
PF1-]	All

CHANGE INDENTATION/REVERSE

Key	Keypad Mode
PF1- < **IND REV**	EDT LK201, EDT VT100
PF1-[	All

DECwindows Interface Equivalent

CHANGE INDENTATION{/FORWARD /REVERSE}

Pull-down menu: Edit > Indentation . . .

Related Commands

ENTER TAB

SET INDENTATION

TAB

UNTAB

CHANGE TEXT_ENTRY_MODE

CHANGE TEXT_ENTRY_MODE — Switches the mode of text-entry in the current buffer between insert and overstrike.

Format

CHANGE TEXT_ENTRY_MODE

Description

The CHANGE TEXT_ENTRY_MODE command switches the mode of the current buffer between insert and overstrike. The status line displays the current text-entry mode.

Keypad Equivalent

Key	Keypad Mode
F14 INSERT **OVERSTR**	EDT LK201, EVE LK201
Ctrl/A	All
ENTER **INSERT OVERSTR**	EVE VT100

DECwindows Interface Equivalent

Buffer status line{Insert Overstrike Unmodifiable}

Related Commands

SET INSERT

SET OVERSTRIKE

CHANGE WINDOW_MODE

CHANGE WINDOW_MODE — Switches between reducing and increasing the number of windows displayed on the screen.

Format

CHANGE WINDOW_MODE

Description

The CHANGE WINDOW_MODE command changes the number of windows displayed on the screen. If the screen has one window, this command creates a second window. If the screen has two or more windows, the CHANGE WINDOW_MODE command reduces the screen display to a single window containing the current buffer.

Keypad Equivalent

Key	Keypad Mode
PF1-equal sign (=)	All

Related Commands

ONE WINDOW

SET SCREEN WINDOW

CHECK LANGUAGE

CHECK LANGUAGE — Analyzes the definitions associated with a language and reports errors.

Format

CHECK LANGUAGE language-name

Qualifiers	Defaults
/DEFINITIONS	/DEFINITIONS
/HELP_INTERFACE	/DEFINITIONS

Qualifiers

/DEFINITIONS (D)

Specifies that the CHECK LANGUAGE command report the following:

- Undefined tokens

- Undefined placeholders
- Unreferenced placeholders
- Package routines with the same name as placeholders
- Package parameters with the same name as tokens
- Parameters defined with the same name in multiple packages
- Routines defined with the same name in multiple packages
- Invalid topic strings

/HELP_INTERFACE

Specifies that the CHECK LANGUAGE command report invalid topic strings.

Parameter

language-name

Specifies the name of the language whose definitions are to be checked. Wildcards are not permitted.

Restrictions

The /DEFINITIONS and /HELP_INTERFACE qualifiers are mutually exclusive.

Description

The CHECK LANGUAGE command analyzes the definitions associated with a language. This command detects and reports the following:

- Undefined tokens – An undefined token has not been defined by a DEFINE TOKEN command, but appears in a menu placeholder body.
- Undefined placeholders – An undefined placeholder has not been defined by a DEFINE PLACEHOLDER command. It appears in the body of a token, or in the body of a nonterminal or menu placeholder; or it appears as the value of a /PLACEHOLDER qualifier on a DEFINE TOKEN or DEFINE PLACEHOLDER command.
- Unreferenced placeholders – An unreferenced placeholder has been defined using a DEFINE PLACEHOLDER command. It does not appear in the body of any token, or in the body of any nonterminal or menu placeholder, and is not used as the value of a /PLACEHOLDER qualifier on a DEFINE TOKEN or DEFINE PLACEHOLDER command.
- Package routines with the same name as placeholders – A token name that is the same as the routine name in a package associated with the language prevents LSE from accessing the template for the routine.
- Package parameters with the same name as tokens – A placeholder name that is the same as the parameter name in a package associated with the language prevents the DECTPU procedure associated with the parameters for that package from being called to properly define the placeholders for the parameter. It can cause incorrect behavior and erroneous messages.

- Parameters defined with the same name in multiple packages – A parameter name that is defined in multiple packages associated with the language might cause the wrong DECTPU procedure to be called for the parameter. It can cause incorrect behavior and erroneous messages. A parameter is not reported as defined in multiple packages if the packages have been defined with the same DECTPU procedure for parameter expansion;that is, the same value on the /PARAMETER_EXPAND qualifier on DEFINE PACKAGE commands.
- Routines defined with the same name in multiple packages – A routine that is defined in multiple packages associated with the language prevents LSE from accessing some of the routine templates, because it will expand only the first definition for a routine that it encounters.
- Invalid topic strings – Topic strings are specified as values on the /TOPIC qualifier on the DEFINE TOKEN and DEFINEPLACEHOLDER commands. A topic string is invalid if there is no corresponding HELP text in the HELP library for the language.

If LSE detects any of these conditions, they are reported in the $CHECK_LANGUAGE buffer, which is displayed in an editing window. You can use the WRITE command to write the contents of this buffer to a file. If LSE does not detect any of the conditions listed previously, a success message is displayed.

Example

```
$ LSEDIT /NODISPLAY /NOCURRENT_FILE /INITIALIZATION=SYS$INPUT: Return
CHECK LANGUAGE/HELP_INTERFACE my_language Return
WRITE /BUFFER=$CHECK_LANGUAGE CHECK_LANGUAGE.LIS Return
QUIT Return
$
```

The size and structure (key depth) of the HELP library and the number of tokens and placeholders that have /TOPIC qualifiers determine the amount of time required to check the HELP library for the language. This checking can take a significant amount of time. It may be more convenient to check the HELP library for a language from a batch procedure.

CLOSE BUFFER

CLOSE BUFFER — Writes and deletes the current buffer.

Format

CLOSE BUFFER

Description

The CLOSE BUFFER command writes the buffer, contingent on buffer attributes and status, then deletes the buffer. If the buffer has the WRITE attribute, and you have modified the contents of the buffer since they were last written, LSE first writes the contents of the buffer to its associated file. If a file is not associated with the buffer, LSE prompts you for a file name.

DECwindows Interface Equivalent

Pop-up menu: User buffer > Close

Pull-down menu: File > Close File

Related Commands

WRITE

CMS

CMS — Invokes VSI Code Management System (CMS) to enable any valid CMS command to execute from within LSE.

Format

CMS [cms-command]

Parameter

cms-command

Specifies any valid specification for a CMS command, including valid qualifiers.

Description

The CMS command invokes CMS from within LSE to let you enter any valid CMS command. To use this command, you must have CMS installed on your system.

Related Commands

GOTO FILE

GOTO SOURCE

READ

REPLACE

RESERVE

SET CMS

UNRESERVE

Examples

1. LSE> ***CMS SET LIBRARY DISK$:[USER.CMSLIB]***

   Sets the specified library as your current CMS library. There after, LSE file-manipulation commands, such as GOTO FILE, GOTO SOURCE, and READ,access that library when you enter a SET CMS command.

2. LSE> ***CMS SHOW RESERVATIONS***

   Reports on the reservation history of all elements in the library set as your current CMS library.

COLLAPSE

COLLAPSE — Compresses text at the current cursor position.

Format

COLLAPSE

Qualifier	Defaults
/DEPTH=n	/DEPTH=1

Qualifier

/DEPTH=n
/DEPTH=1 (D)

Compresses the text at the current cursor position up `n` levels. If you specify the value ALL, this qualifier compresses the text at the cursor position as much as possible.

Note that when you use the COLLAPSE command in query buffers, this command does not support the /DEPTH qualifier.

Description

The COLLAPSE command displays an overview of the text at the current cursor position. Low-level detail lines are replaced by a single overview line. The cursor position is recorded before the text is collapsed for use with future EXPAND commands.

The editor determines the relative level of detail of a line by comparing the indentation of the line with the indentation of other lines. The editor's treatment of the indentation of a line is influenced by indentation adjustment definitions. For more information,see the DEFINE ADJUSTMENT command.

In an SCA query buffer, if the cursor is positioned on a symbol that has been expanded, or on an occurrence within an expansion, the COLLAPSE command causes the occurrences to disappear.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl \	All

DECwindows Interface Equivalent

COLLAPSE

Pop-up menu: Query buffer → Collapse

Pull-down menu: View → Collapse

COLLAPSE/DEPTH=ALL

Pull-down menu: View → Collapse All

Related Command

DEFINE ADJUSTMENT

DEFINE LANGUAGE/OVERVIEW_OPTIONS

EXPAND

FOCUS

MODIFY LANGUAGE

SET NOOVERVIEW

SET OVERVIEW

VIEW SOURCE

COMPILE

COMPILE — Lets you compile the contents of a buffer without leaving LSE.

Format

COMPILE [command-string]

Qualifier	Defaults
/[NO]REVIEW	/NOREVIEW

Qualifier

/REVIEW
/NOREVIEW (D)

Tells LSE whether to wait for the spawned subprocess to complete and then to automatically review any errors reported by the compiler. If you do not specify this qualifier when compiling, you can use the REVIEW command to display any errors after compilation.

By default, the COMPILE command completes as soon as compilation starts. Specifying the /REVIEW qualifier causes the review process to occur as soon as compilation completes.

Parameter

command-string

Specifies the DCL command line to be executed. If you do not specify a command string, LSE uses the command string specified in the definition of the language associated with the current buffer (see the /COMPILE_COMMAND qualifier of the DEFINELANGUAGE command).

If you specify a dollar sign ($) as the first argument on the COMPILE command, LSE replaces the dollar sign with the default COMPILE command. With this feature, you can append file specifications or command qualifiers to the default COMPILE command without having to type the entire command yourself.

If the command string or the string specified on the /COMPILE_COMMAND qualifier contains LSE $FILE, LSE forms the command used to compile the buffer by substituting for LSE$FILE the file specification that corresponds to the buffer. With this feature, you can insert text on the command line immediately after the file specification and before the /DIAGNOSTICS qualifier. If the COMPILE command does not contain LSE$FILE, LSE appends the file specification to the string specified on the qualifier.

Description

The COMPILE command compiles the contents of a buffer without leaving LSE. When you enter this command, LSE writes the current buffer and other buffers associated with the current language back to their files, if they have been modified since they were last written. A buffer is not written if it is designated READ_ONLY.

LSE then forms a DCL command line by appending the file specification of the current buffer to the command string given on the COMPILE command.

If the current buffer's language has diagnostic capabilities (see the DEFINE LANGUAGE/ CAPABILITIES command), LSE appends the /DIAGNOSTICS qualifier to the DCL command it forms, as follows:

`/DIAGNOSTICS=current-device:[current-directory]filespec.DIA`

LSE then spawns a subprocess to execute the DCL command line.

If you specified the /REVIEW qualifier on the COMPILE command, LSE waits for the subprocess to finish executing the DCL command. Otherwise, the COMPILE command completes as soon as the subprocess begins executing the DCL command.

When the subprocess completes, LSE displays a message in the message buffer. If you specified the / REVIEW qualifier, LSE enters review mode and reviews any compilation errors that occurred.

DECwindows Interface Equivalent

Pull-down menu: Source > Compile

Examples

1. `LSE> ***COMPILE $/DEBUG***`

   Compiles the contents of the current buffer. If that buffer is named PROG.FOR,and the current directory specification is USER$:[SMITH], then the following DCL command executes:

   ```
   $ FORTRAN/DEBUG PROG.FOR/DIAGNOSTICS=USER$:[SMITH]PROG.DIA
   ```

   With this command, suppose you had previously specified the following:

   ```
   DEFINE LANGUAGE/COMPILE_COMMAND="FORTRAN 'LSE$FILE'+XXX"
   ```

The DCL command that executes would be as follows:

```
$ FORTRAN PROG.FOR+XXX/DEBUG /DIAGNOSTICS=USER$:[SMITH]PROG.DIA
```

2. LSE> ***COMPILE FORTRAN 'LSE$FILE'+YYY***

   Compiles the contents of the current buffer. If that buffer is named PROG.FOR,and the current directory specification is USER$:[SMITH], the following DCL command executes:

   ```
   $ FORTRAN PROG.FOR+YYY /DIAGNOSTICS=USER$:[SMITH]PROG.DIA
   ```

   Note that the /DIAGNOSTICS qualifier is appended to the Fortran command if you specified that qualifier in the DEFINE LANGUAGE/CAPABILITIES command.

3. ```
   $ OPEN/WRITE X BCOMP.COM
   $ WRITE X "
   $ FORTRAN ",P1," ",P2
   $ CLOSE X
   $ SUBMIT/NOPRINT/DELETE BCOMP
   $ SYNCHRONIZE BCOMP
   ```

   This example is a command procedure, named FORTBATCH.COM, that you could submit as a batch job to compile a Fortran program.

   To submit this as a batch job to the Fortran compiler, enter the following command:

   LSE> ***COMPILE @FORTBATCH***

   If the current buffer is A.FOR and it contains a Fortran program, LSE writes A.FOR to the disk and spawns a subprocess to execute the following DCL command:

   ```
   $ @FORTBATCH A.FOR;2 /DIAGNOSTICS=DISK$:[USER]A.DIA
   ```

   This causes the FORTBATCH procedure to create the file BCOMP.COM, which contains the following DCL compilation command:

   ```
   $ FORTRAN A.FOR;2 /DIAGNOSTICS=DISK$:[USER]A.DIA
   ```

   The FORTBATCH procedure then submits BCOMP.COM to run in batch mode.

   The DCL command SYNCHRONIZE (on the final line of the sample command procedure)causes the subprocess to wait until the batch job completes before it returns control to LSE. This is essential if you are specifying the COMPILE/REVIEW command. LSE considers the compilation to be completed when the subprocess finishes executing. If you do not specify the SYNCHRONIZE command upon completion of a batch job, you cannot use the COMPILE/REVIEW command. However, you can use the LSE REVIEW command to enter Review mode after the batch job finishes the compilation.

CONTINUE

CONTINUE — Ends command entry and returns control to keypad-mode editing.

Format

CONTINUE
   ```

Description

With the CONTINUE command, you can return to keypad editing from the command prompt. You can also press Ctrl/Z at the LSE> prompt to return to keypad editing.

Keypad Equivalent

Ctrl/Z, at the LSE> prompt

Related Commands

DO

CONVERT LIBRARY

CONVERT LIBRARY — Converts the specified library from Version 3. *n* format to Version 4. *n* format.

Format

CONVERT LIBRARY directory-spec1 [directory-spec2]

Parameters

directory-spec1

Specifies the directory specification of the Version 3. *n* library to be converted.

directory-spec2

Specifies the directory in which the Version 4. *n* library is to be created. If this parameter is omitted, the new library is created in the directory specified by *directory-spec1*, and the old library is deleted. If this parameter is specified and is different from *directory-spec1*, the old library is not deleted.

Description

The CONVERT LIBRARY command converts a Version 3. *n* library to a format compatible with Version 4. *n*. Because Version 4. *n* libraries can contain much more information than Version 3. *n* libraries, it is recommended that you recompile and load new libraries rather than convert libraries, if possible. This command does not apply to the OpenVMS Alpha product.

Related Commands

CREATE LIBRARY

LOAD

Example

```
LSE> SCA CONVERT LIBRARY SCA$:[USER.V3LIB] SCA$:[USER.V4LIB]
```

Uses the existing library [USER.V3LIB] to create a new Version 4. *n* library named [USER.V4LIB].

CREATE LIBRARY

CREATE LIBRARY — Allocates and initializes OpenVMS library files in a specified directory. The new library then becomes an active SCA library.

Format

CREATE LIBRARY directory-spec[,. . .]

Qualifiers	Defaults
/AFTER=[library-spec]	
/BEFORE=[library-spec]	
/[NO]LOG	/LOG
/MODULES=module-count	/MODULES=25
/[NO]REPLACE	/NOREPLACE
/SIZE=block-count	/SIZE=1000

Qualifiers

/AFTER=[library-spec]

Instructs SCA to insert the new library or libraries into the list of active SCA libraries following the library you specify as the value of the qualifier. If you do not specify a value, SCA adds the library or libraries to the end of the list.

/BEFORE=[library-spec]

Instructs SCA to insert the new library or libraries into the list of active SCA libraries in front of the library you specify as the value of the qualifier. If you do not specify a value, SCA adds the library or libraries to the beginning of the list.

/LOG (D)
/NOLOG

Indicates whether SCA reports the successful creation of a library and the resulting list of active libraries.

/MODULES=module-count
/MODULES=25 (D)

Specifies an estimated number of modules in the library.

/REPLACE
/NOREPLACE (D)

Indicates whether LSE replaces an existing library with a new (empty) library.

/SIZE=block-count
/SIZE=1000 (D)

Specifies an estimated size for a library.

Parameter

directory-spec[, . . .]

Specifies one or more directories in which library files are to be allocated and initialized.

Description

The CREATE LIBRARY command initializes a library and defines it as the active library in your current SCA session. When you subsequently invoke SCA, it uses the logical name SCA$LIBRARY to reestablish the active library list.

Related Commands

CONVERT LIBRARY

LOAD

SET LIBRARY

Examples

1. `$ SCA CREATE LIBRARY SCA$:[USER.SCA]/REPLACE`

   Initializes a library in the named directory. SCA replaces the existing library with empty library files.

2. `$ SCA CREATE LIBRARY TOP`

   Initializes a library in the directory defined by logical name TOP.

   For additional examples, see the section about creating a library in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

CUT

CUT — Moves or copies the selected range to the specified buffer.

Format

CUT

Qualifiers	Defaults
/[NO]APPEND	/NOAPPEND
/BUFFER=buffer-name	/BUFFER=$PASTE (D)
/CLIPBOARD	See Description
/[NO]ERASE	/ERASE
/REPLACE	
/SUBSTITUTE	

Qualifiers

/APPEND
/NOAPPEND (D)

Indicates whether the moved text should be appended to the current contents of the receiving buffer, or should replace the current contents of the receiving buffer.

/BUFFER=buffer-name
/BUFFER=$PASTE (D)

Specifies the buffer to receive the text being moved. If the /REPLACE or/SUBSTITUTE qualifier is specified, the specified buffer supplies text to replace text being erased from the current buffer.

/CLIPBOARD

Specifies that the DECwindows clipboard should be used to receive the text being moved, instead of a buffer. The /CLIPBOARD and /BUFFER qualifiers are mutually exclusive.

/ERASE (D)
/NOERASE

Specifies whether the moved text should be deleted from the current buffer. LSE ignores this qualifier if the current buffer is not modifiable.

/REPLACE

Erases the selected text and replaces it with the contents of the specified buffer.

/SUBSTITUTE

Erases the search string, replaces it with the contents of the buffer specified in the /BUFFER qualifier, and finds the next occurrence of the string. To use this qualifier, do the following:

1. Enter the SET SELECT_MARK command (press the SELECT key) at the command prompt.
2. Type the new text in the buffer.
3. Enter the CUT command (press the CUT or REMOVE key) at the command prompt. This places the text in the specified buffer.
4. Enter the SEARCH command (press the FIND key) at the command prompt,followed by the text you want to search for and replace.
5. Press the ENTER key.
6. Enter the CUT/SUBSTITUTE command (press the SUBS key) at the command prompt.

Subsequently, each time you enter the CUT/SUBSTITUTE command, LSE makes one substitution and finds the next occurrence of the search string.

Description

The CUT command removes or copies text within the selected range and moves it into a designated buffer or default location (the DECwindows Clipboard or character-cell $PASTE buffer). The selected

range is the text between the select marker (seethe SET SELECT_MARK command) and the current cursor position. If no select marker has been set and the cursor is positioned on the current search string,that string is moved to the buffer.

The /REPLACE and /SUBSTITUTE qualifiers are mutually exclusive; also, these qualifiers cannot be used in conjunction with the /APPEND and /ERASE qualifiers.

For users of the DECwindows interface, the default setting is /CLIPBOARD;otherwise, the default is /BUFFER=$PASTE.

CUT

Key	Keypad Mode
KP6 **CUT**	EVE LK201, EDT LK201, EDT VT100
E3 **REMOVE**	EDT LK201, EVE LK201
KP8 **REMOVE**	EVE VT100

CUT/APPEND

Key	Keypad Mode
KP9 **APPEND**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

CUT/NOERASE

Key	Keypad Mode
PF1-/E3 **COPY**	EDT LK201, EVE LK201

CUT/REPLACE

Key	Keypad Mode
PF1-KP9 **REPLACE**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

CUT/SUBSTITUTE

Key	Keypad Mode
PF1-Enter **SUBS**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

DECwindows Interface Equivalent

CUT/CLIPBOARD

Pop-up menu: User buffer > Cut

Pull-down menu: Edit > Cut

CUT/NOERASE/CLIPBOARD

Pop-up menu: User buffer > Copy

Pull-down menu: Edit > Copy

Related Commands

PASTE

SET SELECT_MARK

SUBSTITUTE

Example

```
LSE> CUT/BUFFER=TEMP.TXT
```

Places the text being moved in the buffer TEMP.TXT.

DCL

DCL — Executes a DCL command from within your editing session.

Format

DCL dcl-command

Parameter

dcl-command

Specifies the DCL command to be executed. If you do not specify a command, LSE prompts for one. Pressing Ctrl/Z at the prompt cancels the operation.

LSE splits the window to show the DCL buffer. You can edit the DCL buffer to move the output from the DCL command into another buffer. You can use the ONE WINDOW command to remove the DCL window.

Description

The LSE command DCL executes a DCL command from within your editing session. LSE spawns a subprocess for the DCL command you specify and creates a buffer named DCL to contain the output from the command.

Example

```
LSE> DCL DIRECTORY *.TXT
```

Splits the screen and displays the DCL command DIRECTORY and its output (the directory listing) in the second window. The cursor remains in the first window.

DEFINE ADJUSTMENT

DEFINE ADJUSTMENT — Defines the behavior of the LSE viewing commands on individual lines of a source file.

Format

DEFINE ADJUSTMENT adjustment-name [pattern]

Qualifiers	Defaults
/[NO]COMPRESS	/COMPRESS
/[NO]COUNT	/COUNT
/CURRENT=number	/CURRENT=0
/[NO]INHERIT=inherit-keyword	/NOINHERIT
/LANGUAGE=language-name	Current buffer language
/[NO]OVERVIEW	/OVERVIEW
/[NO]PREFIX= (indentation-value,	
adjustment-value)	/NOPREFIX
/SUBSEQUENT=number	/SUBSEQUENT=0
/[NO]UNIT	/NOUNIT

Qualifiers

/COMPRESS (D)
/NOCOMPRESS

Avoids compressing groups and overrides indentation. If a group of lines begins with a /NOCOMPRESS line, the group is never compressed.

/COUNT (D)
/NOCOUNT

Controls whether the matching line contributes to the line count for the group. When determining whether to form a group, the line count is compared with the *minimum_lines* value for the language.

See the description for DEFINE LANGUAGE/OVERVIEW_OPTIONS=MINIMUM_LINES.

/CURRENT=number
/CURRENT=0 (D)

Adjusts the indentation of the current line. If a buffer line matches an adjustment defined with the /CURRENT qualifier, the indentation of the buffer line is adjusted by the number of columns given as the qualifier value. A positive value causes the indentation to be adjusted to the right; a negative value causes the indentation to be adjusted to the left. For example, DEFINE ADJUSTMENT then /CURRENT=1 means “Adjust each line that begins with the word ’ then ’ one column to the right.”

See the DEFINE LANGUAGE/OVERVIEW_OPTIONS=TAB_RANGE description.

/INHERIT=inherit-keyword
/NOINHERIT (D)

Specifies that the indentation for the current line is taken from the adjusted indentation of another line.

You can specify one of the following keywords to determine the indentation of the current line:

Keyword	Description
MAXIMUM	The visible indentation for the current line is taken from the adjusted indentation of either the previous line or the next line, whichever is larger.
MINIMUM	The visible indentation for the current line is taken from the adjusted indentation of either the previous line or the next line, whichever is smaller.
NEXT	The visible indentation for the current line is taken from the adjusted indentation of the next line.
PREVIOUS	The visible indentation for the current line is taken from the adjusted indentation of the previous line.

You cannot specify the /INHERIT qualifier with either the /PREFIX or /SUBSEQUENT qualifier.

/LANGUAGE=language-name

Specifies the language associated with the indentation adjustment. By default, the new adjustment is associated with the language for the current buffer. If there is no language associated with the current buffer, the /LANGUAGE qualifier is required.

/OVERVIEW (D)
/NOOVERVIEW

Controls whether the text of the line is used as the overview line. If a line matches an adjustment defined with the /NOOVERVIEW qualifier, the text of the line is never used as the overview text for compressed lines. Instead, text from a later line is used as the overview text. The /NOOVERVIEW qualifier is used to prevent uninformative text from appearing in overview lines.

/PREFIX=(indentation-value, adjustment-value)
/NOPREFIX (D)

Provides a way to skip a pattern at the beginning of a line to determine indentation or influence adjustment. The /PREFIX qualifier takes the following pair of values:

indentation-value
adjustment-value

The *indentation-value* is one of the following keywords:

- *CURRENT*—Instructs LSE to use the indentation of the first text in the pattern – the beginning of the prefix.

- *FOLLOWING*—Instructs LSE to use the indentation of the text that follows the prefix. If there is no text after the prefix, use the indentation of the prefix.

The *adjustment-value* is one of the following keywords:

- *CURRENT*—Instructs LSE to use the adjustment qualifier values given on the current definition.
- *FOLLOWING*—Instructs LSE to use the adjustment qualifier values from the definition that matches the text following the prefix. If no text follows the prefix on the current line, LSE uses the qualifier values for a blank line. If /PREFIX has an adjustment value of FOLLOWING, other action qualifiers on the definition are ignored.

The combination (CURRENT,CURRENT) is not useful because it causes both the indentation and the adjustments to be taken from the text at the beginning of the pattern. This is the same as having no prefix at all.

You cannot specify the /PREFIX qualifier with the /INHERIT qualifier.

/SUBSEQUENT=number
/SUBSEQUENT=0 (D)

Adjusts the indentation of lines after the current line. If a buffer line matches an adjustment defined with the /SUBSEQUENT qualifier, the indentation of all lines after the given one are adjusted by the number of columns given as the qualifier value. A positive value causes the indentation to be adjusted to the right; a negative value causes the indentation to be adjusted to the left.

Use the /SUBSEQUENT qualifier for language constructs that denote nesting and have well-defined endpoints. Use a positive value at the beginning of the construct and a negative value at the end.

You cannot specify the /SUBSEQUENT qualifier with the /INHERIT qualifier.

/UNIT
/NOUNIT (D)

Treats consecutive lines as a single unit. If consecutive lines in the buffer match adjustments defined with the /UNIT qualifier and have the same adjusted indentation, the sequence of lines is treated as one group, with the first serving as the overview line. It is not required that all elements of the group match the same adjustment definition; it is only required that the /UNIT qualifier be specified on all the definitions.

Parameters

adjustment-name

Specifies the name of the adjustment being defined.

pattern

Specifies the string that LSE compares against source lines. If no pattern is used, the *adjustment-name* parameter is used. For details about the syntax for pattern strings, see the section about pattern-matching rules in the *Guide to Language-Sensitive Editor for OpenVMS Systems*.

Pattern strings match any string that you can specify directly on the command line. Strings with special characters must be enclosed in quotes (“ ”). Regardless of whether the string is quoted, the comparison is case-insensitive. You must use the “$()” convention to enclose named pattern elements.

Definitions with literal strings take precedence over definitions with predefined patterns.

The following patterns are predefined:

- *COLUMN=(first-column[,last-column])*—Limits the column in which the text can start.

  You can specify either the first column, or both the first and last columns. If you specify both the first and last columns, you must enclose the column values in parentheses. If you do not specify the last column, it takes its default from the first column.

- *IDENTIFIER*—Matches a sequence of identifier characters.

- *LINE_END*—Matches the end of a line, optionally preceded by white space.

- *OPTIONAL_SPACE*—Matches any sequence of spaces and tabs.

- *FORMFEED*—Matches a form-feed character.

- *FORTRAN_COMMENT*—Matches only Fortran comment lines.

- *FORTRAN_FUNCTION*— Matches the first line of any Fortran function subprogram. That is defined to be any line that matches the following pattern:

  ```
  type [*number] FUNCTION
  ```

  where

  ```
  type :==          BYTE
                |   LOGICAL
                |   INTEGER
                |   REAL
                |   DOUBLE PRECISION
                |   COMPLEX
                |   DOUBLE COMPLEX
                |   CHARACTER
  NUMBER :==       {DIGIT}...
                |   (*)
  ```

- *PREFIX*—The preceding part of the pattern is a prefix.

- *NUMBER*—Matches any sequence of digits. White space cannot appear between digits. In the case of a match with both NUMBER and IDENTIFIER, NUMBER takes precedence.

Description

The DEFINE ADJUSTMENT command defines the behavior of the LSE viewing commands on individual lines of a source file. With the DEFINE ADJUSTMENT command, you can modify the behavior of overviews to match your formatting conventions. You can save DEFINE ADJUSTMENT commands in your environment file.

Related Commands

COLLAPSE

DEFINE LANGUAGE/OVERVIEW_OPTIONS

DELETE ADJUSTMENT

EXPAND

EXTRACT ADJUSTMENT

FOCUS

SHOW ADJUSTMENT

VIEW SOURCE

Examples

1. LSE> ***DEFINE ADJUSTMENT then /CURRENT=1***

   Adjusts each line that starts with the word *then*one column to the right.

2. LSE> ***DEFINE ADJUSTMENT "$(identifier):" /INHERIT=NEXT***

   Specifies that a line starting with any identifier followed by a colon takes the indentation from the following line.

DEFINE ALIAS

DEFINE ALIAS — Lets you assign an abbreviated sequence of characters to represent a longer string of text. You can then use the EXPAND command to produce the longer string each time the cursor is at the end of the abbreviated sequence.

Format

DEFINE ALIAS alias-name [value]

Qualifiers	Defaults
/INDICATED	
/LANGUAGE=language-name	

Qualifiers

/INDICATED

Instructs LSE to interpret the contiguous sequence of characters before and after the cursor as the alias (long form) for an alias name (short form)that you supply. To specify which characters are valid in an alias name for the language you are using, enter a DEFINE LANGUAGE command with the/IDENTIFIER_CHARACTERS qualifier.

When you use the /INDICATED qualifier, you must not specify the value parameter.

/LANGUAGE=language-name

Specifies the language associated with the alias. The default is the language for the current buffer.

Parameters

alias-name

Specifies the name to be defined as an alias. The characters in the alias name must be in the /IDENTIFIER_CHARACTERS string in the DEFINELANGUAGE command.

value

Specifies a quoted string. When you expand the alias, LSE replaces the alias name with the string given by the value parameter. You must not use a value parameter if you specify the /INDICATED qualifier.

Description

With the DEFINE ALIAS command, you can use a shortened name to generate a string of text. You can specify an identifier at the current cursor position as the text you want to generate. After you define an alias name, you can type the alias and then enter the EXPAND command; the text you have assigned to that alias is then displayed.

DEFINE ALIAS/INDICATED

Key	Keypad Mode
PF1-Ctrl/A	All

Keypad Equivalent

Table 2.1. DEFINE ALIAS/INDICATED

Key	Keypad Mode
PF1-Ctrl/A	All

Related Commands

DEFINE LANGUAGE

EXPAND

Example

```
LSE> DEFINE ALIAS/LANGUAGE=FORTRAN lse "The VSI Language-Sensitive Editor"
```

Causes the quoted string to appear when you type *lse* and then enter the EXPAND command when you are in the Fortran language environment.

See the section about defining an alias in the *Guide to Language-Sensitive Editor for OpenVMS Systems* for additional examples.

DEFINE COMMAND

DEFINE COMMAND — Defines a user command or an abbreviation for an LSE command.

Format

DEFINE COMMAND command-name value-string

Parameters

command-name

Specifies the name to be defined as a command. A command name can contain up to 255 characters, but must begin with a letter, an underscore, or a dollar sign. After the first character, you can use any combination of alphanumeric characters, underscores, or dollar signs.

value-string

Specifies a quoted string containing an LSE command or the leading portion of an LSE command.

Description

With the DEFINE COMMAND command, you can define your own commands or specify an abbreviation for an LSE command. Before the command executes, LSE substitutes the specified value string for the command name.

To define a command for a sequence of commands, use the DO command inside the value string.

Related Commands

CALL

DO

Example

```
LSE> DEFINE COMMAND CLS "DO/TPU ""ERASE(CURRENT_BUFFER)"""
```

Associates the command name CLS with the command DO/TPU “ERASE(CURRENT_BUFFER)”. After entering this command,whenever you type CLS at the command prompt, LSE uses DECTPU to clear all text from the current buffer.

DEFINE KEY

DEFINE KEY — Binds an LSE command to a key.

Format

DEFINE KEY key-specifier string

Qualifiers	Defaults
/DIALOG	
/[NO]IF_STATE=GOLD	/NOIF_STATE
/LEARN	

Qualifiers	Defaults
/LEGEND=string	See text
/REMARK=(string, . . .)	
/STATE=GOLD	
/TOPIC_STRING=string	/TOPIC_STRING=no_topic

Qualifiers

/DIALOG

Specifies that a dialog box should be used to prompt the user for parameters and qualifier values. The command parameters are optional if this qualifier is specified. If command parameters and qualifiers are specified with the /DIALOG qualifier, the parameters and qualifiers are used to set the initial state of the dialog box.

/IF_STATE=GOLD
/NOIF_STATE (D)

Specifies that the key definition applies only to the GOLD (PF1) state.

/LEARN

Indicates that a sequence of keystrokes, called a learn sequence, defines the command to be bound to a key. You must type the keystroke sequence immediately after the command and end the sequence by specifying the END DEFINE command. If you are using the EVE keypad, Ctrl/R is bound to the END DEFINE command by default. However, you do not have to define a key to be the END DEFINE command to use the DEFINE KEY/LEARN command. When LSE records the learn sequence, the key being defined by the DEFINE KEY/LEARN command binds to the END DEFINE command. Therefore, you can press the key that you are defining to end the learn sequence.

When executing the stored sequence, LSE includes your responses to all prompts, but does not prompt you again for such information as the string for a SEARCH command.

You cannot use a learn sequence to enter a key definition while another key is in the process of being defined by another learn sequence.

/LEGEND=string
/LEGEND=?

Specifies the text that appears in the keypad diagram for this key. The string is centered in the figure for the key, or truncated if the string is too long for the figure.

If you do not specify the /LEGEND qualifier with a string, the default is /LEGEND=?.

/REMARK=(string, …)

Specifies the explanatory text displayed when you enter a SHOW KEY/FULL command.

/STATE=GOLD

Moves the functionality of the GOLD (PF1) key to the named key. You cannot specify the string parameter with the /STATE=GOLD qualifier.

/TOPIC_STRING=string
/TOPIC_STRING=no_topic (D)

Specifies the string that the editor uses to retrieve help text for this key for display through the HELP /KEYPAD command.

If you do not specify a string with the /TOPIC_STRING qualifier,the default is /TOPIC_STRING=no_topic.

Parameter

key-specifier

Specifies a keyword that indicates the key to be defined. If you use the DEFINE KEY command to change the definition of a key that was previously defined, LSE does not save the previous definition.

Table 2.2, "LSE Keynames for the Editing and Auxiliary Keypad" lists the LSE key names and their VT200-type (or higher) and VT100-type counterparts for the editing and auxiliary keypad. *Table 2.3, "LSE Keynames for Keys on the Main Keyboard"* lists the LSE keynames and their VT200-type (or higher) and VT100-typecounterparts for the main keyboard keys.

As an alternative to using the /IF_STATE=GOLD qualifier, the key-specifier parameter accepts keynames prefixed with GOLD/. In addition, you can specify control keys as Ctrl/ *x*, where *x* is an alphabetic character (A through Z).

string

Specifies an LSE command to be executed when the key is pressed. This is a required parameter unless you use the /LEARN qualifier; you cannot use the string parameter with either the /LEARN or /STATE=GOLD qualifier.

Table 2.2. LSE Keynames for the Editing and Auxiliary Keypad

Keyname	VT200-Type (or higher)	VT100-Type
PF1	PF1	PF1
PF2	PF2	PF2
PF3	PF3	PF3
PF4	PF4	PF4
KP0,KP1, …,KP9	KP0,KP1, …,KP9	KP0, KP1, …,KP9
PERIOD	Keypad period (.)	Keypad period (.)
COMMA	Keypad comma (,)	Keypad comma (,)
MINUS	Keypad minus (–)	Keypad minus (–)
ENTER	Enter	Enter
UP	↑	↑
DOWN	↓	↓
LEFT	←	←
RIGHT	→	→
E1	Find/E1	

Keyname	VT200-Type (or higher)	VT100-Type
E2	Insert Here/E2	
E3	Remove/E3	
E4	Select/E4	
E5	Prev Screen/E5	
E6	Next Screen/E6	
HELP	Help/F15	
DO	Do/F16	
F7,F8, …,F20	F7,F8 …,F20	

Table 2.3. LSE Keynames for Keys on the Main Keyboard

Keyname	VT200-Type (or higher)	VT100-Type
TAB_KEY	Tab	Tab
RET_KEY	Return	Return
DEL_KEY	Delete	Delete
LF_KEY	LF/F13	Line feed
BS_KEY	BS/F12	Backspace
SPACE_KEY	Space bar	Space bar
CTRL_A_KEY	Ctrl/A	Ctrl/A
CTRL_B_KEY	Ctrl/B	Ctrl/B
.	.	.
.	.	.
.	.	.
CTRL_Z_KEY	Ctrl/Z	Ctrl/Z
NULL_KEY	Ctrl/Space bar	Ctrl/Space bar
FS_KEY	Ctrl/ \	Ctrl/ \
GS_KEY	Ctrl/]	Ctrl/]
RS_KEY	Ctrl/~	Ctrl/~
US_KEY	Ctrl//	Ctrl//

If you want to define a key to be lowercase, you must put the key specifier in lowercase and in quotes. However, GOLD and Ctrl sequences are not case-sensitive. For example, Ctrl/A and Ctrl/a produce the same results. Also, GOLD/A is the same as GOLD/a.

The following combinations of the Ctrl key and keyboard keys can be defined,but unless your terminal has the PASSALL characteristic set, you cannot execute your definitions of these keys:

Ctrl/C	Ctrl/O	Ctrl/Q	Ctrl/S
Ctrl/T	Ctrl/X	Ctrl/Y	

If the following combinations of the Ctrl key and keyboard keys are redefined,the new definition also affects the keyboard key corresponding to that combination. For example, if Ctrl/I is redefined, the TAB key also assumes that new definition.

Combination	Description
Ctrl/I	Tab
Ctrl/M	Carriage return
Ctrl/J	Line feed
Ctrl/H	Backspace

If the first key pressed in response to the (Key:) prompt is a key that does not correspond to a printing key character, LSE echoes the corresponding keyname. Tables *Table 2.2, "LSE Keynames for the Editing and Auxiliary Keypad"* and *Table 2.3, "LSE Keynames for Keys on the Main Keyboard"* list keys that do not correspond to a printable character.

If the first key pressed is the GOLD key, LSE waits for you to press a second key. LSE then echoes the key specifier for the key sequence. For example, if you press the GOLD key and then press the P key, LSE echoes GOLD/P.

Only the first key you press in response to the prompt (or the first two keys if the GOLD key is first) is handled in this special way. Subsequent input to the prompt is treated as though you typed in the text that LSE echoes. Ctrl/C, Ctrl/Z, and the Return key required to end the input line are all handled in this way. Erasing all the text at the prompt (using Ctrl/U or the DELETE command) causes LSE to interpret the next key input as the first key.

Description

The DEFINE KEY command associates an LSE command with a key. You can bind commands to control keys, numeric keypad keys, and the arrow keys on all keyboards. You can also bind a command to the sequence of the GOLD key followed by any keyboard key, where the GOLD key is the key defined to set the GOLD state (usually PF1). (On the VT200-series (or higher) keyboard, you can also bind to the function (F) keys and the keys on the editing keypad.)

The HELP/KEYPAD command uses the values of the /LEGEND and /TOPIC qualifiers to build a keypad diagram for the keypad keys and to access help text for the keys. The SHOW KEY/FULL command displays the strings associated with the /LEGEND, /TOPIC_STRING, and /REMARK qualifiers.

The effect of a key can vary with its context. The DEFINE KEY command provides only for definitions for keys that are used in the work region.

DECwindows Interface Equivalent

DEFINE KEY/DIALOG

Pull-down menu: Show → Show Key *

Related Commands

END DEFINE

Example

```
LSE> DEFINE KEY "GOLD/KP5" "GOTO TOP"
```

If the PF1 key sets the GOLD state, then the key sequence PF1-KP5 always issues a GOTO TOP command after you assign this definition.

See the section about defining keys in the *Guide to Language-Sensitive Editor for OpenVMS Systems* for additional examples.

DEFINE KEYWORDS

DEFINE KEYWORDS — Defines the specified keyword list.

Format

```
DEFINE KEYWORDS keyword-list-name
keyword [/DESCRIPTION=text]
    .
    .
    .
keyword [/DESCRIPTION=text]
END DEFINE
```

Qualifier	Defaults
/DESCRIPTION=text	

Qualifier

/DESCRIPTION=text

Indicates the text to be associated with the individual keyword.

Parameters

keyword-list-name

Identifies the keyword list. The name must follow the rules applied to token names in LSE. You can then use the name as the value you specify for the /KEYWORDS qualifier to the DEFINE TAG command, as well as the parameter for the DELETE KEYWORDS,EXTRACT KEYWORDS, and SHOW KEYWORDS commands.

keyword

Names an individual keyword. Each keyword on the list must appear on a line by itself. You cannot use continuation characters between the lines for each keyword, but you can use a continuation character between a particular keyword and its associated qualifier.

Related Commands

DEFINE TAG

DELETE KEYWORDS

EXTRACT KEYWORDS

SHOW KEYWORDS

Example

```
DEFINE KEYWORDS author_names
   "Pat Jones"   /DESCRIPTION="Project Leader"
   "Chris Brown"
   "Leslie Green"
END DEFINE
```

Creates a keyword list named author_names and lists the individual names.

DEFINE LANGUAGE

DEFINE LANGUAGE — Specifies the characteristics of a language.

Format

DEFINE LANGUAGE language-name

Qualifiers	Defaults
/BOOK=file-spec, defined_language	
/CAPABILITIES=[NO]DIAGNOSTICS	/CAPABILITIES=NODIAGNOSTICS
/COMMENT=(specifier, …)	
/COMPILE_COMMAND=string	
/EXPAND_CASE=AS_IS	/EXPAND_CASE=AS_IS
/EXPAND_CASE=LOWER	/EXPAND_CASE=AS_IS
/EXPAND_CASE=UPPER	/EXPAND_CASE=AS_IS
/FILE_TYPES=(file-type[, …])	
/FORTRAN=[NO]ANSI_FORMAT	/FORTRAN=NOANSI_FORMAT
/[NO]HELP_LIBRARY=file-spec	/NOHELP_LIBRARY
/IDENTIFIER_CHARACTERS=string	
/INITIAL_STRING=string	
/LEFT_MARGIN= *n*	/LEFT_MARGIN=1
/LEFT_MARGIN=CONTEXT_DEPENDENT	
/OVERVIEW_OPTIONS= (MINIMUM_LINES= *m*,	
TAB_RANGE= (t1,t2))	
/PLACEHOLDER_DELIMITERS=	
(delimiter-specification[, …])	See text
/PUNCTUATION_CHARACTERS=string	/PUNCTUATION_CHARACTERS= ",;() "
/[NO]QUOTED_ITEM= (QUOTES=string	
[,ESCAPES=string])	/NOQUOTED_ITEM
/REFERENCE=book reference, defined_language	
/RIGHT_MARGIN= *n*	/RIGHT_MARGIN=80
/TAB_INCREMENT= *n*	/TAB_INCREMENT=4

Qualifiers	Defaults
/TOPIC_STRING=string	
/VERSION=string	
/[NO]WRAP	/NOWRAP

Qualifiers

/CAPABILITIES=DIAGNOSTICS
/CAPABILITIES=NODIAGNOSTICS (D)

Specifies whether the compiler can generate diagnostic files.

/BOOK=file-spec, defined_language

Specifies the default online-book file name, defining the book LSE uses to retrieve online text for a placeholder or token whose book is undefined.

/COMMENT=(specifier, …)

Specifies the character sequences of comments in the language. The specifiers are as follows:

- ASSOCIATED_IDENTIFIER=keyword

  Indicates the preferred association of comments to identifier. You can specify one of the following values:

  - NEXT – Indicates that comments should be associated with the next identifier.
  - PREVIOUS – Indicates that comments should be associated with the preceding identifier.

- BEGIN=list of quoted strings

  END=list of quoted strings

  Defines the character sequences that start and end bracketed comments. A bracketed comment begins and ends with explicit comment delimiters. (Note that the beginning and ending comment delimiters can be the same, but need not be.) The list provided with the specifiers BEGIN and END can be any of the following:

  - A string that is the one open comment sequence for the language. You must enclose this in quotes.
  - A parenthesized list of strings, each one of which can be an open comment sequence for the language. You must enclose each one in quotes.

  The list accompanying the BEGIN specifier must be consistent with the list accompanying the END specifier. If the BEGIN specifier lists a string, the END specifier must also list a string.

  Bracketed comments are recognized by the formatting commands (see the ALIGN and FILL commands) and placeholder operations (see the ERASEPLACEHOLDER command and the /DUPLICATION qualifier of the DEFINEPLACEHOLDER command).

- TRAILING=list of quoted strings Defines the character sequence that introduces line-oriented comments. Aline-oriented comment begins with a special character sequence(consisting of

one or more characters) and ends at the end of the line. The list provided with the TRAILING specifier can be any of the following:

- A string that is the one-line comment sequence for the language.
- A list of strings enclosed in parentheses; each string can be a line-comment sequence for the language.

Line comments are recognized by the formatting commands and placeholder operations, just as bracketed comments are.

- LINE=list of quoted strings

  Requires that the comment delimiter be the first character that is not blank on the line. The LINE specifier is particularly useful with block comments, such as the following:

```
/*
** Here is the inside of a comment
** which has LINE="**" specified
*/
```

- FIXED=quoted string, column number

  Used for languages that require that a specific comment delimiter be placed in a specific column, such as FIXED=("* ",1) for COBOL.

Note that for the specifier you cannot use any character that you used in the /PLACEHOLDER *delimiter-specification*.

/COMPILE_COMMAND=string

Specifies the default command string for the COMPILE command. (See the explanation of the *command-string* parameter in the COMPILE command entry.)

/EXPAND_CASE=AS_IS (D)
/EXPAND_CASE=LOWER
/EXPAND_CASE=UPPER

Specifies the case of the text of the inserted template. The value AS_IS specifies that the inserted template be expanded according to the case in the token or placeholder definition. The values LOWER and UPPER specify that the inserted template be expanded in lowercase or uppercase, respectively.

/FILE_TYPES=(file-type[, …])

Specifies a list of file types that are valid for the language being defined. The file types must be enclosed in quoted strings. When LSE reads a file into a buffer, it sets the language for that buffer automatically if it recognizes the file type. For example, a Fortran file type (.FOR) sets the language to Fortran. Note that the period character must be included with the file type.

/FORTRAN=ANSI_FORMAT
/FORTRAN=NOANSI_FORMAT (D)

Specifies special processing for ANSI Fortran. Note that some commands behave differently when you use the /FORTRAN qualifier. Specifying NOANSI_FORMAT causes LSE to insert templates in non-ANSI (tab) format.

/HELP_LIBRARY=file-spec
/NOHELP_LIBRARY (D)

Specifies the HELP library where you can find help text for placeholders and tokens defined in this language. LSE applies the default file specification SYS$HELP:HELPLIB.HLB. If you want to access some HELP library other than SYS$HELP, you must supply an explicit device name.

/IDENTIFIER_CHARACTERS=string

Specifies the characters that can appear in token and alias names in that language. This list of characters is used in various contexts for the/INDICATED qualifier.

The list of identifier characters also determines what LSE considers to be a word. A word is a sequence of identifier characters, possibly followed by one or more blanks. All nonblank, nonidentifier characters are considered to be distinct words.

If you do not specify the /IDENTIFIER_CHARACTERS qualifier, LSE supplies the following values by default:

```
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ%$_0123456789"
```

/INITIAL_STRING=string

Specifies the initial text to appear in a newly created buffer.

/LEFT_MARGIN= *n*
/LEFT_MARGIN=1 (D)
/LEFT_MARGIN=CONTEXT_DEPENDENT

Specifies the left margin setting to be associated with the language.

If you specify CONTEXT_DEPENDENT as the column number, LSE uses the indentation of the current line to determine the left margin when you use the /WRAP qualifier. When you use the FILL command, LSE uses the indentation of the first line of each selected paragraph to determine the left margin.

/OVERVIEW_OPTIONS=(MINIMUM_LINES=m, TAB_RANGE= (t1,t2))

Specifies both the minimum number of lines an overview line must hide and the range of acceptable tab increments.

The specifiers are as follows:

- MINIMUM_LINES= *m*

  Specifies the minimum number of lines an overview line must hide. The default is 1. For example, if the value of the parameter on MINIMUM_LINES is 5, a line hides other lines only if there are at least five lines to hide. This specifier helps the user to avoid having very small source-line groups,and thus to avoid many expansion levels.

- TAB_RANGE=(t1,t2)

  The TAB_RANGE specifier indicates the range of tab values for which the adjustment definitions are valid. The default is (4,8). The second value must be at least twice the first value; both values must be positive. For example, if the tab range is (4,8), LSE assumes that the adjustment definitions will work for any DEFINE LANGUAGE/TAB_INCREMENT value from

4 to 8, inclusive. If you specify a /TAB_INCREMENT value outside the tab range, then LSE recomputes indentation to make the adjustments work.

For best performance, it is recommended that you avoid recomputation by choosing a range that covers reasonable values. The numbers specified for the DEFINE ADJUSTMENT/CURRENT and DEFINE ADJUSTMENT/SUBSEQUENT commands must work for any tab increment value in the tab range.

/PLACEHOLDER_DELIMITERS=(delimiter-specification[, …])

Specifies starting and ending strings that delimit placeholders. Placeholders can specify single constructs or lists of constructs. The delimiters for each type of placeholder are specified as a pair of quoted strings separated by commas and enclosed in parentheses.

The format of a delimiter specification is as follows:

```
keyword=(starting-string,ending-string)
```

Possible keywords are REQUIRED, REQUIRED_LIST, OPTIONAL, OPTIONAL_LIST, or PSEUDOCODE. If you do not use the PSEUDOCODE keyword, the default is NOPSEUDOCODE. The maximum length of these strings is seven characters.

The following is an example of a complete set of placeholder delimiter specifications:

```
/PLACEHOLDER_DELIMITERS = ( -
 REQUIRED =("{<",">}"), -
 REQUIRED_LIST=("{<",">}..."), -
 OPTIONAL =("[<",">]"), -
 OPTIONAL_LIST=("[<",">]..."), -
 PSEUDOCODE=("«" , "»"))
```

If any of the five keywords are not specified with the /PLACEHOLDER_DELIMITERS qualifier, LSE applies the following defaults:

```
/PLACEHOLDER_DELIMITERS = ( -
 REQUIRED =("{","}"), -
 REQUIRED_LIST=("{","}..."), -
 OPTIONAL =("[","]"), -
 OPTIONAL_LIST=("[","]..."), -
 NOPSEUDOCODE)
```

The following table lists the placeholder delimiters accepted by each compiler.

Language	Placeholder Delimiters
Ada	`{ }, { }..., [ ], [ ]..., « »`
VSI BASIC	`{ }, { }..., [ ], [ ]..., « »`
VAX BLISS-32	`{~ ~}, {~ ~}... , [~ ~], [~ ~]..., « »`
VSI C	`{@ @}, {@ @}..., [@ @], [@ @]..., « »`
VSI COBOL	`{ }, { }..., [ ], [ ]..., « »`
VSI C++	`{@ @}, {@ @}..., [@ @], [@ @]..., « »`
F90	`{ }, { }..., [ ], [ ]...,`

Language	Placeholder Delimiters
VSI Fortran	`{ }, { }..., [ ], [ ]..., « »`
VSI Pascal	`%{ }%, %{ }%..., %[ ]%, %[ ]%..., « »`
PL/I	`{ }, { }..., [ ], [ ]..., « »`

Note that for the specifier you cannot use any character that you used in the /COMMENT *specifier*.

/PUNCTUATION_CHARACTERS=string
/PUNCTUATION_CHARACTERS= ",; ( ) " (D)

Specifies the characters that are considered punctuation marks, or delimiters,in the language. When a placeholder name and its enclosing brackets are deleted, preceding white space is also deleted if there are punctuation characters to delimit the program constructs.

/QUOTED_ITEM=(QUOTES=string [,ESCAPES=string])
/NOQUOTED_ITEM (D)

Describes the syntax of certain language elements, such as strings, that require special handling for proper text formatting. LSE uses the /QUOTED_ITEM qualifier to detect comments properly. LSE does not acknowledge comment strings that occur within quoted items, nor does it acknowledge quoted elements that occur within comments.

The value of the /QUOTED_ITEM qualifier indicates the syntax of a quoted item. This value must be a keyword list. The keywords are as follows:

- QUOTES

  This keyword is required and must have an explicit value. The value must be a quoted string denoting all the quote characters in the language. LSE assumes that quoted items begin and end with the same character.

- ESCAPES

  This keyword is optional. If given, the value is required and must be a quoted string containing the escape characters for quoted items. Some languages use escape characters to insert quote characters into strings. For example, C uses the backslash (\) as an escape character. If you omit this keyword, LSE assumes that the language inserts quote characters into strings by doubling them.

/REFERENCE=book_reference, defined_language

Specifies the book-reference tag string, defining the section of a book to display for a placeholder or token whose reference tag is undefined.

/RIGHT_MARGIN=n
/RIGHT_MARGIN=80 (D)

Specifies the right margin setting to be associated with the language. By default, the right margin is set at column 80.

/TAB_INCREMENT=n
/TAB_INCREMENT=4 (D)

Specifies that tab stops be set every *n* columns, beginning with column 1.

/TOPIC_STRING=string

Specifies a prefix string to be concatenated to the /TOPIC_STRING qualifier specified in a placeholder or token definition before LSE looks up the help text for that placeholder or token. (Typically, this is the name of the language in the HELP library.)

/VERSION=string

Specifies a string that represents the version number of the tokens and placeholders associated with this language. Use the SHOW LANGUAGE command to display this string.

/WRAP
/NOWRAP (D)

Specifies whether the ENTER SPACE command (bound to the space bar by default) should wrap text when there is too much to fit on the current line. The /NOWRAP qualifier disables text wrapping.

Parameter

language-name

Specifies the name of the language whose characteristics are to be defined.

Description

The DEFINE LANGUAGE command specifies a language so LSE can properly recognize language-specific text characteristics.

After you specify these language characteristics by using the DEFINELANGUAGE command, you can use the MODIFY LANGUAGE command when you want to make subsequent changes.

Related Commands

DELETE LANGUAGE

EXTRACT LANGUAGE

MODIFY LANGUAGE

SET LANGUAGE

SHOW LANGUAGE

Examples

```
1.  DEFINE LANGUAGE ADA -
      /CAPABILITIES=DIAGNOSTICS -
      /COMPILE_COMMAND="ADA" -
      /FILE_TYPES=(.ADA) -
      /IDENTIFIER_CHARACTERS= -
          "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
$_0123456789" -
```

```
      /INITIAL_STRING="{compilation_unit}" -
      /COMMENT=(TRAILING="-") -
      /PLACEHOLDER_DELIMITERS=( -        REQUIRED=("{","}"), -
        REQUIRED_LIST=("{","}..."), -
        OPTIONAL=("[","]"), -
        OPTIONAL_LIST=("[","]...") ) -
      /PUNCTUATION_CHARACTERS=",;()*.'" -
      /QUOTED_ITEM=(QUOTES="""'") -
      /TAB_INCREMENT=4 -
      /TOPIC_STRING="ADA Language_Topics"
```

Defines characteristics of the Ada language.

```
2.  DEFINE LANGUAGE PASCAL -
      /CAPABILITIES=DIAGNOSTICS -
      /COMMENT=(BEGIN=("{","(*"),END=("}","*)")) -
      /COMPILE_COMMAND="PASCAL " -
      /FILE_TYPES=(.PAS) -
      /IDENTIFIER_CHARACTERS= -
          "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ$
%_0123456789" -
      /INITIAL_STRING="%{compilation_unit}%" -
      /PLACEHOLDER_DELIMITERS=( -
        REQUIRED=("%{","}%"), -
        REQUIRED_LIST=("%{","}%..."), -
        OPTIONAL=("%[","]%"), -
        OPTIONAL_LIST=("%[","]%...") ) -
      /PUNCTUATION_CHARACTERS=",;:()[]{}.'" -
      /QUOTED_ITEM=(QUOTES="""'") -
      /TAB_INCREMENT=4 -
      /TOPIC_STRING="PASCAL "
```

Defines characteristics of the Pascal language.

See the sections about language definition in the *Guide to Language-Sensitive Editor for OpenVMS Systems* for additional examples.

DEFINE PACKAGE

DEFINE PACKAGE — Defines a subroutine package for which subroutine call templates are automatically generated.

Format

DEFINE PACKAGE package-name

Qualifiers	Defaults
/HELP_LIBRARY=file-spec	
/LANGUAGE=(language [, …])	
/PARAMETER_EXPAND=TPU_procedure_prefix	
/ROUTINE_EXPAND=TPU_procedure_prefix	

Qualifiers	Defaults
/TOPIC_STRING=string	

Qualifiers

/HELP_LIBRARY=file-spec

Specifies the HELP file (.HLB file) where you can find help text for placeholders and tokens defined for this package. If omitted, no HELP file is associated with the package.

LSE applies the default file specification SYS$HELP:HELPLIB.HLB. If you want to access some device or directory other than SYS$HELP, you must supply an explicit device and directory name.

/LANGUAGE=(language [, …])

Specifies the languages from which LSE can use package entries. If you do not specify a language, LSE uses the language of the current buffer. If no language is associated with the current buffer, an error occurs.

/PARAMETER_EXPAND=TPU_procedure_prefix

Lets you customize calling sequences. Normally, LSE uses a fixed algorithm to produce the appropriate placeholder definitions from the DEFINEPARAMETER command. If the default algorithm is inadequate, you can supply explicit DECTPU procedures to produce the DEFINE PLACEHOLDER command that you want.

The argument provided with the /PARAMETER_EXPAND qualifier must be the prefix of a DECTPU procedure name. Whenever LSE expands a parameter, it concatenates this prefix and the current language name,and looks for a DECTPU procedure by that name. For details, see Appendix D.

/ROUTINE_EXPAND=TPU_procedure_prefix

Lets you customize calling sequences. LSE normally uses a fixed algorithm to produce the appropriate token and placeholder definitions from the DEFINE ROUTINE command. If the default algorithm is inadequate, you can supply explicit DECTPU procedures to produce the DEFINE PLACEHOLDER or DEFINE TOKEN commands that you want.

The argument provided with the /ROUTINE_EXPAND qualifier must be the prefix of a DECTPU procedure name. Whenever LSE expands a routine, it concatenates this prefix and the current language name,and looks for a DECTPU procedure by that name. For example, if you specify /ROUTINE_EXPAND=my_routine_expand_ and the current language is Fortran, LSE looks for a DECTPU procedure named my_routine_expand_fortran. For details, see Appendix D.

/TOPIC_STRING=string

Specifies a prefix string to be concatenated to the TOPIC_STRING specified for tokens and placeholders associated with the routine or parameter definitions. If omitted, the null string is used as the topic string. LSE uses the topic string to look up help text for the package.

Parameter

package-name

Specifies the name of the package being defined.

Description

The DEFINE PACKAGE command defines a subroutine package for which subroutine-call templates are automatically generated. Packages can contain routine definitions, which describe calls to subroutines, and parameter definitions, which describe parameters for subroutine calls.

Related Commands

DEFINE PARAMETER

DEFINE ROUTINE

DELETE PACKAGE

EXTRACT PACKAGE

SHOW PACKAGE

Example

```
DEFINE PACKAGE system_services -
    /LANGUAGES =(BASIC,C,COBOL,FORTRAN,PLI) -
    /HELP_LIBRARY = HELPLIB -
    /TOPIC_STRING = "system_services" -
    /ROUTINE_EXPAND = "LSE$PKG_EXPAND_ROUT_" -  ! Special routines for
    /PARAMETER_EXPAND = "LSE$PKG_EXPAND_PARM_"  ! system services

DEFINE ROUTINE sys$add_holder -
    /PACKAGE = system_services -
    /DESCRIPTION = "Add Holder Record To The Rights Database" -
     id/BY_VALUE, -
     holder/BY_REFERENCE, -
     attrib/BY_VALUE/OPTIONAL -
DEFINE PARAMETER id -
    /PACKAGE = system_services -
DEFINE PARAMETER holder -
    /PACKAGE = system_services -
DEFINE PARAMETER attrib -
    /PACKAGE = system_services -
```

Shows the incorporation of the DEFINE PACKAGE command in a complete package definition, along with the DEFINE PARAMETER and DEFINE ROUTINE commands.

DEFINE PARAMETER

DEFINE PARAMETER — Defines a parameter within a package.

Format

DEFINE PARAMETER

Qualifier	Defaults
/PACKAGE=package-name	

Qualifier

/PACKAGE=package-name

Specifies the name of the package with which the parameter is associated.

Parameter

param-name

Specifies the name of the parameter. This name must be unique among the tokens of any language from which the package is used.

Description

The DEFINE PARAMETER command defines a parameter within a package. That parameter can be associated with more than one routine by means of the DEFINEROUTINE command.

Related Commands

DEFINE PACKAGE

DEFINE ROUTINE

DELETE PARAMETER

EXPAND

EXTRACT PARAMETER

SHOW PARAMETER

Example

```
DEFINE PACKAGE system_services -
    /LANGUAGES =(BASIC,C,COBOL,FORTRAN,PLI) -
    /HELP_LIBRARY = HELPLIB -
    /TOPIC_STRING = "system_services" -
    /ROUTINE_EXPAND = "LSE$PKG_EXPAND_ROUT_" -  ! Special routines for
    /PARAMETER_EXPAND = "LSE$PKG_EXPAND_PARM_"  ! system services

DEFINE ROUTINE sys$add_holder -
    /PACKAGE = system_services -
    /DESCRIPTION = "Add Holder Record To The Rights Database" -
     id/BY_VALUE, -
     holder/BY_REFERENCE, -
     attrib/BY_VALUE/OPTIONAL -

DEFINE PARAMETER id -
    /PACKAGE = system_services -
DEFINE PARAMETER holder -
    /PACKAGE = system_services -
DEFINE PARAMETER attrib -
    /PACKAGE = system_services -
```

Shows the incorporation of the DEFINE PARAMETER command into a complete package definition, along with the DEFINE PACKAGE and DEFINE ROUTINE commands.

DEFINE PLACEHOLDER

DEFINE PLACEHOLDER — Creates a placeholder for use with a specific language and establishes the characteristics of that placeholder.

Format

```
DEFINE PLACEHOLDER placeholder-name

placeholder body

END DEFINE

or

DEFINE PLACEHOLDER placeholder-name

/PLACEHOLDER= other-placeholder
```

Qualifiers	Defaults
/[NO]AUTO_SUBSTITUTE	/NOAUTO_SUBSTITUTE
/BOOK=string	
/DESCRIPTION=string	
/DUPLICATION=specifier	/DUPLICATION=
	CONTEXT_DEPENDENT
/LANGUAGE=language-name	
/LEADING=string	
/PLACEHOLDER=other-placeholder	
/[NO]PSEUDOCODE	/PSEUDOCODE
/REFERENCE=book_reference	
/SEPARATOR=string	
/TOPIC_STRING=string	
/TRAILING=string	
/TYPE=type-specifier	/TYPE=NONTERMINAL

Qualifiers

/AUTO_SUBSTITUTE
/NOAUTO_SUBSTITUTE (D)

Specifies whether you want the next placeholder with this name to be replaced with the same text you typed over the current placeholder.

/BOOK=string

Specifies an online book file name that LSE uses to retrieve the online text for a placeholder.

/DESCRIPTION=string

Specifies a single line of text to be displayed along with the placeholder name when the placeholder name appears in a menu during an EXPAND operation.

/DUPLICATION=specifier
/DUPLICATION=CONTEXT_DEPENDENT (D)

Specifies the type of duplication to be performed when the placeholder is duplicated (either by expanding it or by typing over it). The specifier is one of the following keywords:

- CONTEXT_DEPENDENT

  If the placeholder is the only item within its segment (that is, if it is either the only item before or the only item within a trailing comment), LSE duplicates it vertically (see the VERTICAL keyword in this list). Otherwise, LSE duplicates it horizontally. White space can precede or follow the placeholder.

- HORIZONTAL

  LSE places the duplicate immediately to the right of the original. If you specify a separation string, LSE places the string between the original and the duplicate.

- VERTICAL

  LSE places the duplicate on the next line immediately under the original. If a separation string is specified, LSE places it at the end of the original. If the original placeholder is in the commented segment of the line, LSE also duplicates the comment delimiters directly underneath the delimiters in the original line. If necessary, LSE adds close comment delimiters to the original line to close a bracketed comment on that line.

/LANGUAGE=language-name

Specifies the language associated with the placeholder. By default, the new placeholder is defined for use with the current buffer's language.

/LEADING=string

Specifies any leading text to be associated with the placeholder. The ERASEPLACEHOLDER command recognizes this text and erases it along with the placeholder. The leading text must not have any trailing blank spaces,because the ERASE PLACEHOLDER command always skips over such spaces.

/PLACEHOLDER=other-placeholder

Specifies the name of another defined placeholder from which this placeholder inherits its definition. A placeholder defined with the /PLACEHOLDER qualifier cannot be named on the /PLACEHOLDER qualifier of any other definition. The /PLACEHOLDER qualifier is mutually exclusive with all other qualifiers except the /LANGUAGE qualifier.

/PSEUDOCODE (D)
/NOPSEUDOCODE

Specifies whether pseudocode can be entered at a specific placeholder. If you specify the /NOPSEUDOCODE qualifier for a placeholder,that placeholder cannot be used with pseudocode.

/REFERENCE=book_reference, defined_language

Specifies the book-reference tag string, which defines the section of a book to display for a placeholder.

/SEPARATOR=string

Specifies the string that separates each duplication of the placeholder. See the description of the /DUPLICATION qualifier.

/TOPIC_STRING=string

Specifies a quoted string that LSE uses to retrieve help text for this placeholder. This string is appended to the string you specify with the/TOPIC_STRING qualifier of the DEFINE LANGUAGE command to form the complete string of topics that LSE uses for looking up the help text for this placeholder.

/TRAILING=string

Specifies any trailing text to be associated with the placeholder. The ERASEPLACEHOLDER command recognizes this text and erases it along with the placeholder. The trailing text must not have any leading blank spaces because the ERASE PLACEHOLDER command always skips over such spaces.

/TYPE=type-specifier
/TYPE=NONTERMINAL (D)

Specifies the kind of placeholder being defined. The type specifier can be NONTERMINAL, MENU, or TERMINAL.

Parameters

placeholder-name

Specifies the name of the placeholder being defined. A placeholder name must be unique within a language and can be a quoted string. To redefine an existing placeholder, you must first delete it using the DELETE PLACEHOLDER command.

placeholder body

Is the body of the placeholder being defined. The interpretation of the placeholder body depends on the type of placeholder. LSE displays the body of a terminal placeholder when you attempt to expand the placeholder. Note that displaying this text does not replace the terminal placeholder and its delimiters.

The body of a nonterminal placeholder is the text of the placeholder expansion; when a nonterminal placeholder is expanded, the placeholder name and enclosing delimiters are replaced with the text of the placeholder body.

A nonterminal placeholder can have more than one quoted string in each body line. For the expansion of the placeholder, you can set the indentation of each string by using the /INDENTATION qualifier and its associated keywords.

Each quoted string in the body line of a nonterminal placeholder can take the qualifier and keywords described in the following section.

Nonterminal Body Qualifier	Defaults
/INDENTATION=(keyword1 [,integer1, keyword2])	

keyword1

You can specify any of the following options for *keyword1*:

Option	Description
EXPAND	Indents the string to the column of the first character of the nonterminal placeholder being expanded. This is the default value if the first body line is not a null string.
CURRENT	Indents the string to the indentation of the line containing the placeholder or token. This is the default value if the first body line is a null string.
PREVIOUS	Indents the string to the indentation of the line before the line containing the placeholder or token.
FIXED	Indents the string to the specified column.

integer1

You can specify any integer for the *integer1* option. The default is 0. The integer is added to the column position as specified by *keyword1* and adjusts the indentation by that number of columns. The integer can be negative. When the value for *keyword1* is FIXED, *integer1*specifies the column position at which to put body text; it must be positive.

keyword2

You can specify either of the following options for *keyword2*:

Option	Description
TAB	Specifies that *integer1* should be interpreted as specifying an adjustment in terms of tab increments rather than columns. *Integer1* is multiplied by the tab increment for the buffer before it is added to the column specified by *keyword1*.
SPACE	Specifies that *integer1* should be interpreted as specifying an adjustment in terms of spaces. This is the default.

Note that you cannot specify *keyword2* when *keyword1* has a FIXED value.

If there is more than one quoted string in a body line, a comma must separate the strings. For Fortran, if the body line is inside of a comment or there is a tab in the body lines,the /INDENTATION qualifier and associated keywords do not take effect for the first quoted string for each body line.

For more information about using the /INDENTATION qualifier,see the examples for the EXPAND command.

Each line of the body of a menu placeholder represents one option in the menu. An option can be a string of text, placeholder name, or token name. If the option is a string of text, it must appear in quotes. If the option is a placeholder name or a token name and does not appear in quotes,that placeholder name or token name appears in uppercase letters in the menu display. For a placeholder name or token name to appear in lowercase letters in a menu, you must enter the placeholder name or token name as a lowercase, quoted string.

Each line in the body of a menu placeholder can take one or more of the following qualifiers:

Menu Body Qualifiers	**Default**
/DESCRIPTION=string	
/[NO]LIST	/NOLIST
/PLACEHOLDER	
/TOKEN	

/DESCRIPTION=string

Specifies a description string displayed in the right-hand column of the menu. If this qualifier is omitted, LSE gets the description string from the corresponding definition if the line has either the /TOKEN or the/PLACEHOLDER qualifier. If neither /TOKEN nor /PLACEHOLDER is specified, the line is a literal string and the value of the /DESCRIPTION string defaults to the empty string.

/LIST

/NOLIST (D)

Specifies whether the delimiters for the placeholder should be list delimiters. Use this qualifier only in conjunction with the /PLACEHOLDER qualifier.

/PLACEHOLDER

Specifies that the name or string is the name of a placeholder in the language. This qualifier is mutually exclusive with the /TOKEN qualifier.

/TOKEN

Specifies that the name or string is the name of a token in the language. This qualifier is mutually exclusive with the /PLACEHOLDER and /[NO]LIST qualifiers.

Description

The DEFINE PLACEHOLDER command creates and establishes the characteristics of a placeholder for use with a particular language. A placeholder definition consists of a DEFINE PLACEHOLDER command followed by a placeholder body (which might occupy more than one line). If you do not specify the /PLACEHOLDER qualifier, you must end the placeholder body with an END DEFINE command.

Subsequently, you can use the new placeholder with the EXPAND and HELP/LANGUAGE commands.

Related Command

DEFINE TOKEN

DELETE PLACEHOLDER

END DEFINE

EXPAND

EXTRACT PLACEHOLDER

HELP/INDICATED

SHOW PLACEHOLDER

Examples

1. ```
   DEFINE PLACEHOLDER parameter -
         /LANGUAGE = EXAMPLE -
         /DESCRIPTION = "Parameter name"
         /DUPLICATION = HORIZONTAL -
         /SEPARATOR = ", " -
         /TYPE = TERMINAL
         "A string of letters and digits starting with a letter."
       END DEFINE
   ```

   Creates a placeholder named parameter and establishes its characteristics.

2. ```
   DEFINE PLACEHOLDER "#IF" -
         /LANGUAGE=C -
         /TYPE=NONTERMINAL -
         "#if {@constant expression@}"/INDENTATION=(FIXED,1)
         "[@#else_clause@]"/INDENT=(FIXED,1)
         "#endif"/INDENTATION=(FIXED,1)
   END DEFINE
   ```

   The /INDENTATION=(FIXED,1) qualifier puts the body text at column 1while the expanding operation is performed. With the definitions in this example, the expanded placeholder [@#IF@] is as follows:

   ```
   #if {@constant expression@}[@#else_clause@]#endif
   ```

   For additional examples, see the sections about placeholder definitions and language elements in the *Guide to Language-Sensitive Editor for OpenVMS Systems*.

DEFINE ROUTINE

DEFINE ROUTINE — Defines templates for a routine contained within a subroutine package.

Format

DEFINE ROUTINE routine-name [parameter, . . .

Qualifiers	Defaults
/BOOK=string	
/DESCRIPTION=string	

Qualifiers	Defaults
/PACKAGE=package-name	
/REFERENCE=string	
/TOPIC_STRING=string	

Qualifiers

/BOOK/=string

Specifies an online-book file name string that LSE uses to retrieve text for the specified routine.

/DESCRIPTION=string

Specifies a single line of text to be displayed along with the routine name when the routine name appears in a menu during an EXPAND operation. The string is also passed to the /ROUTINE_EXPAND procedure, if any. (The default algorithm for producing routine calls from DEFINE ROUTINE commands does not make use of this value.)

/PACKAGE=package-name

Specifies the name of the package with which the routine is associated. You must specify this qualifier.

/REFERENCE=string

Specifies a reference in an online book file name string that LSE uses to retrieve text for a routine.

/TOPIC_STRING=string

Specifies a quoted string that LSE uses to retrieve help text for this routine.

Parameters

routine-name

Specifies the name of the routine. Routine names must be unique within a package. Furthermore, routine names cannot conflict with any token names used by LSE for any language using the package.

parameter, . . .

Specifies the names of the parameters of the routine. These parameters must be defined (using the DEFINE PARAMETER command) before expanding an instance of a call on this routine. However, the parameters do not need to be defined before the DEFINE ROUTINE command. If you omit this qualifier, the routine is presumed to have no parameters.

The following qualifiers are position-sensitive; they can be used only with the list of parameters to the routine:

- **/BY_VALUE** Indicates that the parameter is passed by value.
- **/BY_REFERENCE** Indicates that the parameter is passed by address.
- **/BY_DESCRIPTOR** Indicates that the address of the parameter descriptor is passed.

- **/[NO]OPTIONAL** Specifies whether the parameter is required or optional. The default is/NOOPTIONAL.

The /BY_VALUE, /BY_REFERENCE, and /BY_DESCRIPTOR qualifiers are mutually exclusive. These qualifiers are used primarily for languages, such as COBOL,that require explicit specification of passing mechanisms for routine calls.

Description

The DEFINE ROUTINE command defines templates for a routine contained within a subroutine package. This command makes the routine known as an element of a package. The first time the routine name is expanded, LSE generates an appropriate template and simulates a corresponding DEFINE TOKEN command. Thus, you can expand and unexpand routines in the same manner as tokens. Note, however, that commands such as SHOW TOKEN do not operate on tokens defined from routines; instead, you should use the appropriate routine commands, such as SHOW ROUTINE and EXTRACT ROUTINE.

Related Command

DEFINE PACKAGE

DEFINE PARAMETER

DELETE ROUTINE

EXPAND

EXTRACT ROUTINE

SHOW ROUTINE

Example

```
DEFINE PACKAGE system_services -
       /LANGUAGES =(BASIC,C,COBOL,FORTRAN,PLI) -
       /HELP_LIBRARY = HELPLIB -
       /TOPIC_STRING = "system_services" -
       /ROUTINE_EXPAND = "LSE$PKG_EXPAND_ROUT_" -  ! Special routines for
       /PARAMETER_EXPAND = "LSE$PKG_EXPAND_PARM_"  ! system services

   DEFINE ROUTINE sys$add_holder -
       /PACKAGE = system_services -
       /DESCRIPTION = "Add Holder Record To The Rights Database" -
        id/BY_VALUE, -
        holder/BY_REFERENCE, -
        attrib/BY_VALUE/OPTIONAL

    DEFINE PARAMETER id -
       /PACKAGE = system_services

    DEFINE PARAMETER holder -
       /PACKAGE = system_services

    DEFINE PARAMETER attrib -
       /PACKAGE = system_services
```

Shows the incorporation of the DEFINE ROUTINE command into a complete package definition, along with DEFINE PACKAGE and DEFINE PARAMETER commands.

DEFINE TAG

DEFINE TAG — Defines the specified tag.

Format

DEFINE TAG tag-name

Qualifiers	Defaults
/EMPTY=string-list	/EMPTY="None"
/KEYWORDS=keyword-list-name	
/LANGUAGE=language-name	
/SUBTAGS=tag-list	
/TYPE=type-keyword	/TYPE=TEXT

Qualifiers

/EMPTY=string-list
/EMPTY="None" (D)

Specifies one or more strings indicating that a use of the structured tag has no subtags. If you do not specify the /EMPTY qualifier, there will be no way to explicitly indicate that an occurrence of the tag is empty. You can always use implicitly empty tags by starting a new top-level tag after the current top-level tag, or by terminating the comment block.

You use this qualifier only with the /TYPE=STRUCTURED case.

/KEYWORDS=keyword-list-name

Defines the keywords that you can use with this tag. You must specify the *keyword-list-name* parameter by using the DEFINE KEYWORDS command. If you specify /KEYWORDS=*,this indicates that any keyword is allowed and no checking of keywords is to be done.

You use this qualifier only with the /TYPE=KEYWORD case.

/LANGUAGE=language-name

Specifies the language associated with the tag being defined. If you do not specify a language, the default is the language of the current buffer.

/SUBTAGS=tag-list

Indicates the subtags that can appear in a structured tag. The special case /SUBTAGS=* indicates that any tag is allowed. For example, you would use this special case for the PARAMETERS tag.

You use this qualifier only with the /TYPE=STRUCTURED case.

/TYPE=type-keyword

Indicates the type of the tag. You can specify any one of the following types:

Keyword Type	Description
TEXT	Ordinary text tag (default)
KEYWORD	List of keywords to be parsed at compile time
STRUCTURED	Sequence of zero or more subtags

Parameter

tag-name

Specifies the name of the tag being defined. The tag name must consist only of alphanumeric characters, the dollar sign ($), or the underscore (_), and can contain embedded blanks. Tag names are case-insensitive. If you include embedded blanks, place the name inside quotation marks.

Description

The DEFINE TAG command defines the specified tag. Tags are headings embedded inside comments for use with design reports. You can save the definition in an environment file and direct the compiler to process tags with the /DESIGN qualifier.

For more information about how to use tags, seethe section about design information in the *Guide to Language-Sensitive Editor for OpenVMS Systems*.

Related Commands

DELETE TAG

EXTRACT TAG

SHOW TAG

Examples

1. LSE> ***DEFINE TAG "functional description"***

   Defines the tag functional description and indicates that the tag is an ordinary text tag.

2. LSE> ***DEFINE TAG parameters /TYPE=STRUCTURED /SUBTAGS=****
   _LSE> ***/EMPTY=("None", "Omitted")***

   Defines the tag parameters, specifies that the tag type is STRUCTURED, and indicates that any tag is allowed. The /EMPTY=("None", "Omitted") qualifier indicates that you can use either the word None or the word Omitted in your programs to explicitly indicate that the tag has no subtag values.

DEFINE TOKEN

DEFINE TOKEN — Defines an editing token for use with the EXPAND command.

Format

```
DEFINE TOKEN     token-name
```

```
token body

END DEFINE

or

DEFINE TOKEN     token-name

/PLACEHOLDER= placeholder-name
```

Qualifiers	Defaults
/BOOK=string	
/DESCRIPTION=string	
/LANGUAGE=language-name	
/PLACEHOLDER=placeholder-name	
/REFERENCE=string	
/TOPIC_STRING=string	

Qualifiers

/BOOK=string

Specifies an online-book file name that LSE uses to retrieve text for a token.

/DESCRIPTION=string

Specifies some text to be displayed along with the token name when the token name appears in a menu during an EXPAND operation, or in a SHOW TOKEN display.

/LANGUAGE=language-name

Specifies the language associated with the token. By default, the token is defined for use with the current language.

/PLACEHOLDER=placeholder-name

Specifies the name of a defined placeholder that expands in place of the token. The token gets its description, topic string, and body from the defining placeholder.

Note that the /PLACEHOLDER qualifier is mutually exclusive with the /DESCRIPTION and/ TOPIC_STRING qualifiers, and the END DEFINE command must not be used on the DEFINE TOKEN command when /PLACEHOLDER is specified. No token body is specified with the / PLACEHOLDER qualifier.

/REFERENCE=string

Specifies a book-reference tag string, defining the section of a book to display for a token.

/TOPIC_STRING=string

Specifies a quoted string that LSE uses to retrieve help text for this token. This string is appended to the /TOPIC_STRING qualifier specified in the DEFINE LANGUAGE command to form the complete string of topics that LSE uses to look up the help text for this token.

Parameter

token-name

Specifies the name for the token being defined. Each token for a particular language must have a unique name. Token and alias names must not conflict. A token name can be any character including a blank space, but not a leading or trailing space.

token body

Is the text of the token expansion. When the token is expanded, the token name is replaced with the text of the token body. A token can have more than one quoted string in each body line. For the expansion of the token, you can set the indentation of each string by using the /INDENTATION qualifier and its associated keywords.

Each quoted string in the body line of a token can take the qualifier and keywords described as follows.

Nonterminal Body Qualifier	**Defaults**
/INDENTATION=	
(keyword1 [,integer1, keyword2])	

keyword1

You can specify any of the following options for *keyword1*:

Option	**Description**
EXPAND	Indents the string to the column of the first character of the nonterminal placeholder being expanded. This is the default value if the first body line is not a null string.
CURRENT	Indents the string to the indentation of the line containing the placeholder or token. This is the default value if the first body line is a null string.
PREVIOUS	Indents the string to the indentation of the line before the line containing the placeholder or token.
FIXED	Indents the string to the specified column.

integer1

You can specify any integer for the *integer1* option. The default is 0. The integer is added to the column position as specified by *keyword1* and adjusts the indentation by that number of columns. The integer can be negative. When the value for *keyword1* is FIXED, *integer1*specifies the column position at which to put body text and must be positive.

keyword2

You can specify either of the following options for *keyword2*:

Option	**Description**
TAB	Specifies that *integer1* should be interpreted as specifying an adjustment in terms of tab

Option	Description
	increments rather than columns. *Integer1* is multiplied by the tab increment for the buffer before it is added to the column specified by *keyword1*.
SPACE	Specifies that *integer1* should be interpreted as specifying an adjustment in terms of spaces. This is the default.

Note that you cannot specify *keyword2* when *keyword1* has a FIXED value.

If there is more than one quoted string in a body line, a comma must separate the strings. For Fortran, if the body line is inside of a comment or there is a tab in the body lines,the /INDENTATION qualifier and associated keywords do not take effect for the first quoted string for each body line.

For more information about the use of the /INDENTATION qualifier,see the examples for the EXPAND command.

Description

The DEFINE TOKEN command defines an editing token for use with the EXPAND command. When you enter the EXPAND command while the cursor is positioned immediately after the token name or an abbreviation of the token name, LSE replaces the input string with the body of the token.

Related Commands

DELETE TOKEN

EXPAND

EXTRACT TOKEN

SHOW TOKEN

Example

1.
```
DEFINE TOKEN ASSIGNMENT -
        /LANGUAGE = EXAMPLE -
        /DESCRIPTION = "Assignment statement"
        "{identifier} = {expression}"
      END DEFINE
```

   Creates a token named ASSIGNMENT and establishes its characteristics.

2.
```
DEFINE TOKEN Parameter_template
      /PLACEHOLDER = Parameter
```

   Creates a token named Parameter_template. When you expand this token, LSE substitutes the placeholder named Parameter for the token.

3.
```
DEFINE TOKEN { -
      /LANGUAGE=C

      "{"/INDENTATION=EXPAND
```

```
   "[@block declaration@]
..."/INDENTATION=(CURRENT,1,TAB)
   ""/INDENTATION=CURRENT
   "{@statement@}
..."/INDENTATION=(CURRENT,1,TAB)
   "}"/INDENTATION=CURRENTEND DEFINE
```

The /INDENTATION=(CURRENT,1,TAB) qualifier indents the body text at the current indentation plus the number of spaces equivalent to one tab increment for the language. Specifying /INDENTATION=EXPAND indents the body text at the cursor's position. Specifying /INDENTATION=CURRENT replaces the body text at the current indentation level. With these definitions, you can expand the token { in the following example:

```
if (a == b) {
```

It becomes the following:

```
if (a == b) {
    [@block declaration@]...
    {@statement@}...}
```

For additional examples, see the sections about token definitions and defining language elements in the *Guide to Language-Sensitive Editor for OpenVMS Systems*.

DELETE ADJUSTMENT

DELETE ADJUSTMENT — Removes a name from the list of adjustments associated with a language.

Format

DELETE ADJUSTMENT adjustment-name

Qualifier	Defaults
/LANGUAGE=language-name	

Qualifier

/LANGUAGE=language-name

Names the language associated with the adjustment being deleted. By default, LSE deletes the adjustment from the set of adjustments defined for the current language. By using the /LANGUAGE qualifier, you can delete adjustments from other languages as well.

Parameter

adjustment-name

Specifies the name of the adjustment to be deleted.

Description

The DELETE ADJUSTMENT command removes a specified name from the list of adjustments associated with a language.

Related Command

DEFINE ADJUSTMENT

EXTRACT ADJUSTMENT

SHOW ADJUSTMENT

Example

```
LSE> DELETE ADJUSTMENT/LANGUAGE=EXAMPLE then
```

Removes the adjustment named then from the list of adjustments associated with the language EXAMPLE.

DELETE ALIAS

DELETE ALIAS — Deletes the definition of an alias name.

Format

DELETE ALIAS alias-name

Qualifier	Defaults
/LANGUAGE=language-name	

Qualifier

/LANGUAGE=language-name

Specifies the name of the language in which the alias is defined. The default is the current language.

Parameter

alias-name

Specifies the alias name to be deleted.

Description

The DELETE ALIAS command cancels the definition of an alias name established by a previous DEFINE ALIAS command.

Related Commands

DEFINE ALIAS

Examples

```
LSE> DELETE ALIAS lse
```

Cancels the definition of the alias named lse.

DELETE BUFFER

DELETE BUFFER — Deletes a buffer.

Format

DELETE BUFFER [buffer-name]

Parameter

buffer-name

Indicates which buffer is to be deleted. The default is the current buffer.

Description

The DELETE BUFFER command deletes the specified buffer. If the buffer is being displayed, LSE replaces it with another buffer. You cannot delete system buffers.

If the specified buffer has been modified and is not read-only, LSE prompts you to answer Y if you want to continue the DELETE BUFFER operation. Otherwise, answer N.

Related Commands

GOTO BUFFER

GOTO FILE

SHOW BUFFER

Example

```
LSE> DELETE BUFFER USER.BUF
```

Deletes the buffer named USER.BUF.

DELETE COMMAND

DELETE COMMAND — Deletes the definition of the specified user-defined command.

Format

DELETE COMMAND command-name

Parameter

command-name

Specifies the command to be deleted.

Description

The DELETE COMMAND command cancels the definition of a command previously established by a DEFINE COMMAND command.

Related Commands

DEFINE COMMAND

Example

```
LSE> DELETE COMMAND XYZ
```

Cancels the definition of the user-defined command XYZ.

DELETE KEY

DELETE KEY — Deletes the specified key definition.

Format

DELETE KEY key-specifier

Qualifier	Defaults
/[NO]IF_STATE=GOLD	/NOIF_STATE

Qualifier

/IF_STATE=GOLD
/NOIF_STATE (D)

Specifies that the key definition to be deleted is for the GOLD state. The default is to delete the key definition for the default state.

Parameter

key-specifier

Indicates the keyword or single printing character for the key to be deleted. Valid key-specifiers include all keynames recognized by the DEFINE KEY command.

Description

The DELETE KEY command cancels a key definition established by a previous DEFINEKEY command. If the key is a printing key, LSE restores the original function of inserting a printing character at the current cursor position; otherwise, the key is undefined.

Related Commands

DEFINE KEY

SHOW KEY

Example

```
LSE> DELETE KEY KP7
```

Deletes the definition for the KP7 key (the 7 key on the numeric keypad).

DELETE KEYWORDS

DELETE KEYWORDS — Cancels the specified keywords-list definition.

Format

DELETE KEYWORDS keyword-list-name

Parameter

keyword-list-name

Specifies the keyword list to be deleted.

Description

The DELETE KEYWORDS command cancels the keyword list defined by the previous DEFINEKEYWORDS command.

Related Commands

DEFINE KEYWORDS

EXTRACT KEYWORDS

SHOW KEYWORDS

Example

```
LSE> DELETE KEYWORDS author_name
```

Cancels the keyword list named *author_name*.

DELETE LANGUAGE

DELETE LANGUAGE — Cancels the specified language definition.

Format

DELETE LANGUAGE language-name

Parameter

language-name

Specifies the language to be deleted.

Description

The DELETE LANGUAGE command cancels the language defined by the previous DEFINELANGUAGE command. LSE does not actually delete the tokens, placeholders, and aliases associated with the language, but it makes them unavailable for use. If you subsequently enter a DEFINE LANGUAGE command for the same language name, LSE re-associates all the previously defined tokens, placeholders, and aliases with the new language definition. Thus, you can use the DELETE LANGUAGE command as a step in modifying the properties of a language definition.

Related Commands

DEFINE LANGUAGE

EXTRACT LANGUAGE

SHOW LANGUAGE

Example

```
LSE> DELETE LANGUAGE ADA
```

Cancels the previously defined characteristics for the Ada language.

DELETE LIBRARY

DELETE LIBRARY — Deletes an SCA library from an OpenVMS directory.

Format

DELETE LIBRARY library-spec[, . . .]

Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/[NO]LOG	/NOLOG

Qualifiers

/CONFIRM
/NOCONFIRM (D)

Indicates whether the delete function will request a confirmation of the deletion of each library.

To delete an SCA library, you must respond to the confirmation prompt by typing Y, YE, or YES. Otherwise,the library is not deleted.

/LOG
/NOLOG (D)

Indicates whether successful deletion of the SCA libraries will be reported.

Parameter

library-spec[, . . .]

Specifies one or more libraries to be deleted. The library must be one of the current SCA libraries established by the SET LIBRARY command. You can use a library number in place of a library specification. For example, the primary library is library #1. You can also specify a wildcard name expression.

Related Commands

CREATE LIBRARY

LOAD

SET LIBRARY

SHOW LIBRARY

Example

```
LSE> DELETE LIBRARY/CONFIRM SCA$:[USER.SCA]
```

Deletes a library after confirmation that the library should be deleted.

DELETE MODULE

DELETE MODULE — Deletes specified modules of source-analysis data from SCA libraries.

Format

DELETE MODULE module-name[, . . .]

Qualifiers	Defaults
/[NO]CONFIRM	/NOCONFIRM
/DECLARATION_CLASS=declaration-class	
/LIBRARY[=library-spec]	/LIBRARY=primary-library
/[NO]LOG	/LOG

Qualifiers

/CONFIRM
/NOCONFIRM (D)

Tells SCA whether to prompt you to confirm each module deletion.

To delete a module, you must respond to the confirmation prompt by typing Y, YE,or YES. If you specify N, NO, or press Return, SCA does not delete the module. SCA considers any other response to be ambiguous and reissues the confirmation prompt.

/DECLARATION_CLASS=declaration-class

Indicates the class of the module to be deleted. The following declaration classes are supported:

- PRIMARY – Module implementation
- ASSOCIATED – Module specification

If you do not specify a declaration class, SCA deletes both classes, if they exist.

/LIBRARY[=library-spec]
/LIBRARY=primary-library (D)

Specifies an SCA library containing the module to be deleted. This library must be one of the current libraries (established by a SET LIBRARY command).

If you do not specify a library, the primary SCA library is the default; that is, the module is deleted from the first of the current SCA libraries.

/LOG (D)
/NOLOG

Indicates whether SCA reports successful deletion of a module.

Parameter

module-name[, . . .]

Specifies the names of the modules to be deleted from the current library. You can specify a wildcard name expression.

Description

The DELETE MODULE command allows you to selectively update a specific SCA library.

Example

```
$ SCA DELETE MODULE module_1
```

Deletes module_1 from the library.

DELETE OVERVIEW

DELETE OVERVIEW — N/A

Format

DELETE OVERVIEW

Qualifier

/BUFFER=buffer name

Indicates the buffer whose overview information is to be reset. The default is the current buffer.

Description

The DELETE OVERVIEW command gets rid of all the overview lines in the current buffer and makes all the real lines in the buffer visible.

Example

```
LSE> DELETE OVERVIEW
```

DELETE PACKAGE

DELETE PACKAGE — Deletes a package definition without deleting the routines or parameters associated with the package.

Format

DELETE PACKAGE package-name

Parameter

package-name

Names the package definition to be deleted.

Description

The DELETE PACKAGE command deletes the specified package. The routines and parameters associated with the package are not deleted, but they are no longer available for use. If a subsequent DEFINE PACKAGE command is entered for the same package name, all the previously defined routines and parameters become associated with the new package definition. Thus, you can use the DELETE PACKAGE command, followed by the DEFINE PACKAGE command, to modify the properties of a package definition.

Related Commands

DEFINE PACKAGE

EXTRACT PACKAGE

SHOW PACKAGE

Example

```
LSE> DELETE PACKAGE system_services
```

Deletes the package named system_services.

DELETE PARAMETER

DELETE PARAMETER — Deletes a parameter definition from a package.

Format

DELETE PARAMETER parameter-name

Qualifier	Defaults
/PACKAGE=package-name	

Qualifier

/PACKAGE=package-name

Specifies the name of the package containing the parameter to be deleted. The DELETE PARAMETER command requires this qualifier.

Parameter

parameter-name

Specifies the name of the parameter to be deleted.

Description

The DELETE PARAMETER command deletes the specified parameter definition from the package specified by the /PACKAGE qualifier.

Related Commands

DEFINE PARAMETER

EXTRACT PARAMETER

SHOW PARAMETER

Example

```
LSE> DELETE PARAMETER/PACKAGE=system_services id
```

Deletes the parameter named id from the package named system_services.

DELETE PLACEHOLDER

DELETE PLACEHOLDER — Removes a name from the list of placeholders associated with a language.

Format

DELETE PLACEHOLDER name

Qualifier	Defaults
/LANGUAGE=language-name	

Qualifier

/LANGUAGE=language-name

Names the language associated with the placeholder being deleted. By default, LSE deletes the placeholder from the set of placeholders defined for the current language. Using the /LANGUAGE qualifier, you can delete placeholders from other languages as well.

Parameter

name

Specifies the name of the placeholder to be deleted.

Description

The DELETE PLACEHOLDER command removes a specified name from the list of placeholders associated with a language.

Related Commands

DEFINE PLACEHOLDER

EXTRACT PLACEHOLDER

SHOW PLACEHOLDER

Example

```
LSE> DELETE PLACEHOLDER/LANGUAGE=EXAMPLE parameter
```

Removes the placeholder named parameter from the list of placeholders associated with the language EXAMPLE.

DELETE QUERY

DELETE QUERY — Deletes the specified query.

Format

DELETE QUERY [query-name]

Parameter

query-name

Specifies the query to be deleted. If you omit the query name, the current query is deleted. You can specify wildcards.

Description

The DELETE QUERY command deletes an SCA query.

DECwindows Interface Equivalent

Pop-up menu: Query buffer > Delete Query

Related Commands

FIND

GOTO QUERY

NEXT QUERY

PREVIOUS QUERY

SHOW QUERY

Example

```
LSE> DELETE QUERY 1
```

Removes the query named 1.

DELETE ROUTINE

DELETE ROUTINE — Deletes a routine definition from a package.

Format

DELETE ROUTINE routine-name

Qualifier	Defaults
/PACKAGE=package-name	

Qualifier

/PACKAGE=package-name

Indicates the package containing the routine definition to be deleted. The DELETE ROUTINE command requires this qualifier.

Parameter

routine-name

Specifies the name of the routine to be deleted.

Description

The DELETE ROUTINE command deletes a routine definition from a package. If the routine has already been expanded in the current editing session, the tokens defined by the expansion remain.

Related Commands

DEFINE ROUTINE

EXTRACT ROUTINE

SHOW ROUTINE

Example

```
LSE> DELETE ROUTINE/PACKAGE=system_services sys$add_holder
```

Deletes the routine named sys$add_holder from the package named *system_services*.

DELETE TAG

DELETE TAG — Removes a name from the list of tags associated with a language.

Format

DELETE TAG name

Qualifier	Defaults
/LANGUAGE=language-name	

Qualifier

/LANGUAGE=language-name

Names the language associated with the tag being deleted. By default, LSE deletes the tag from the set of tags defined for the current language. Using the /LANGUAGE qualifier, you can delete tags from other languages as well.

Parameter

name

Specifies the name of the tag to be deleted.

Description

The DELETE TAG command removes a specified name from the list of tags associated with a language.

Related Commands

DEFINE TAG

EXTRACT TAG

SHOW TAG

Example

```
LSE> DELETE TAG/LANGUAGE=EXAMPLE parameters
```

Removes the tag named parameters from the list of tags associated with the language EXAMPLE.

DELETE TOKEN

DELETE TOKEN — Removes a token name from the list of tokens associated with a language.

Format

DELETE TOKEN name

Qualifier	Defaults
/LANGUAGE=language-name	

Qualifier

/LANGUAGE=language-name

Specifies the language associated with the token being deleted. By default, LSE deletes the token from the set of tokens defined for the current language. Using the /LANGUAGE qualifier, you can delete tokens from other languages as well.

Parameter

name

Specifies the token name to be deleted.

Description

The DELETE TOKEN command removes a token name from the list of tokens associated with either the current language or a language you specify.

Related Commands

DEFINE TOKEN

EXTRACT TOKEN

SHOW TOKEN

Example

```
LSE> DELETE TOKEN/LANGUAGE=EXAMPLE assignment
```

Removes the token assignment from the list of tokens associated with the language EXAMPLE.

DELETE WINDOW

DELETE WINDOW — Deletes the current window.

Format

DELETE WINDOW

Description

The DELETE WINDOW command deletes the current window, unless there is only one window. The remaining windows are enlarged to occupy the entire screen.

Related Commands

ONE WINDOW

SET SCREEN WINDOW

DO

DO — Directs LSE to execute LSE commands or DECTPU program statements.

Format

DO [string[, . . .]]

Qualifiers	Defaults
/BUFFER[=buffer-name]	
/[NO]CONTINUE	/CONTINUE
/LSE	/LSE
/PROMPT=prompt-string	
/TPU	/LSE

Qualifiers

/BUFFER[=buffer-name]

Indicates that LSE should read commands from the specified buffer and execute the commands or DECTPU program statements within that buffer. The default is the current buffer. If you do not specify either the /BUFFER or /PROMPT qualifier, LSE executes the current buffer.

/CONTINUE (D)
/NOCONTINUE

Indicates whether LSE prompts for a single string to be executed, or for multiple strings to be executed. If you specify the /NOCONTINUE qualifier, LSE repeatedly prompts for additional commands until you enter a CONTINUE command.

You use the /[NO]CONTINUE qualifier with the /PROMPT qualifier; you must not specify the /NOCONTINUE qualifier with the /TPU qualifier.

/LSE (D)

Indicates that the strings are LSE commands.

/PROMPT=prompt-string

Indicates that LSE should prompt you for a command (or DECTPU program statement) to execute.

The /PROMPT and /BUFFER qualifiers are mutually exclusive. If you specify the string parameter, you cannot specify the /PROMPT or /BUFFER qualifier. If you do not specify the /BUFFER qualifier but specify the /PROMPT qualifier, LSE prompts you for a command and does not execute the current buffer.

/TPU

Indicates that the strings are DECTPU program statements. When specifying the /TPU qualifier, you cannot use the /NOCONTINUE qualifier.

Parameter

string[, . . .]

Specifies a list of comma-separated commands or statements to be executed. Commands with embedded spaces, such as GOTO BUFFER, must be enclosed by double quotation marks.

Description

With the DO command, you can enter commands from a command line or from a buffer. You can specify a list of commands to be executed, or direct LSE to prompt you for LSE/SCA commands or DECTPU program statements (see the *DEC Text Processing Utility Reference Manual* for a description of DECTPU programs).

To end the prompting for commands and return to keypad editing, enter the CONTINUE command, or press Ctrl/Z.

Keypad Equivalent

DO/CONTINUE/PROMPT=LSE Command>

Key	**Keypad Mode**
PF1-KP7 **COMMAND**	EDT LK201, EDT VT100, EVE LK201
DO **DO**	EDT LK201, EVE LK201

Key	Keypad Mode
PF4 **DO**	EVE VT100

DO/NOCONTINUE/PROMPT=LSE>

Key	Keypad Mode
Ctrl/Z	All

DO/CONTINUE/TPU/PROMPT=TPU>

Key	Keypad Mode
PF1-Ctrl/Z	All

Related Commands

CALL

CONTINUE

GOTO COMMAND

EXTEND

Examples

1. LSE> ***DO "GOTO LINE","PASTE"***

   Moves the cursor to the end of the line in the current direction and copies the contents of the $PASTE buffer at that position.

2. LSE> ***DO/TPU "ERASE(MESSAGE_BUFFER)"***

   Invokes DECTPU to erase the contents of the message buffer. Any messages that have accumulated at the bottom of your screen are removed.

END DEFINE

END DEFINE — Ends a body of text that begins with a DEFINE command.

Format

END DEFINE

Description

The END DEFINE command ends the body that follows a DEFINE PLACEHOLDER or DEFINE TOKEN command, if the placeholder or token definition has a body. The END DEFINE command ends the list of keywords defined by the DEFINE KEYWORDS command.

The END DEFINE command also ends the sequence of keystrokes that follows a DEFINE KEY/ LEARN command. To use the END DEFINE command for this purpose,enter the command by pressing a key you have defined to be the END DEFINE key.

Related Commands

DEFINE KEY

DEFINE KEYWORDS

DEFINE PLACEHOLDER

DEFINE TOKEN

Example

```
DEFINE PLACEHOLDER parameter -
  /LANGUAGE = EXAMPLE -
  /DESCRIPTION = "Parameter name"
  /DUPLICATION = HORIZONTAL -
  /SEPARATOR = ", " -
  /TYPE = TERMINAL
  "A string of letters and digits starting with a letter."
END DEFINE
```

Shows the position of the END DEFINE command at the end of a DEFINE PLACEHOLDER command.

END REVIEW

END REVIEW — Ends an LSE REVIEW session.

Format

END REVIEW

Description

The END REVIEW command ends the current REVIEW session (initiated by a REVIEW or COMPILE/REVIEW command) and deletes the window containing the$REVIEW buffer.

DECwindows Interface Equivalent

Pop-up menu: Review buffer > End Review

Related Commands

REVIEW

ENLARGE WINDOW

ENLARGE WINDOW — Enlarges the current window.

Format

ENLARGE WINDOW line-count

Parameter

line-count

Specifies the number of screen lines you want to add to the current window. If you do not supply this parameter, LSE prompts you for the number of lines to add.

The maximum size of a window depends on the size and type of the terminal screen you are using. The minimum size is one line of text and one line for the status line.

Description

The ENLARGE WINDOW command enlarges the window the text cursor is in (if you are using more than one window). LSE shrinks the other window (or windows) accordingly.

Related Commands

SHRINK WINDOW

Example

```
LSE> ENLARGE WINDOW 10
```

Adds ten lines to the current window, taking them proportionally from the other window (or windows) on the screen.

ENTER COMMENT

ENTER COMMENT — Converts pseudocode into comments.

Format

ENTER COMMENT

Qualifiers	Defaults
/BLOCK	/BLOCK
/LINE	/BLOCK

Qualifiers

/BLOCK (D)

Specifies that the comment should be entered above the cursor (or selected text range), which formats the comment according to the placeholder LSE$BLOCK_COMMENT.

You cannot specify both the /BLOCK and /LINE qualifiers.

/LINE

Specifies that the comment should be entered at the end of the current line (or selected text range), which formats the comment according to the placeholder LSE$LINE_COMMENT.

You cannot specify both the /LINE and /BLOCK qualifiers.

Description

The ENTER COMMENT command converts pseudocode into comments. It inserts a comment near the current cursor position.

If the cursor is on a pseudocode placeholder, the command moves the placeholder's text into the comment and replaces the placeholder with the LSE$GENERIC placeholder. The cursor is then positioned on the generic placeholder.

If the cursor is in a comment, the LSE editor finds a nearby pseudocode placeholder P, and converts P's content into a comment. The LSE$GENERIC placeholder is inserted in place of P and the cursor remains on the generic placeholder. The command qualifiers are ignored when the cursor is on a comment.

If the cursor is not on a placeholder or comment, the command inserts a new comment and puts the LSE $GENERIC placeholder inside the comment. The cursor is then positioned on the generic placeholder.

If there is a sequence of pseudocode placeholders and a selected range is active when ENTER COMMENT is executed, all text in the selected range is converted into a comment and the placeholders are replaced with LSE$GENERIC placeholders accordingly. The cursor is positioned on the first placeholder after the comment.

The ENTER COMMENT command requires definitions for three placeholders, as follows:

- LSE$BLOCK_COMMENT – Specifies the comment format to be used by ENTER COMMENT/ BLOCK
- LSE$LINE_COMMENT – Specifies the comment format to be used by ENTER COMMENT/LINE
- LSE$GENERIC – Specifies the text to be inserted in place of the pseudocode placeholder removed by ENTER COMMENT

The following are example definitions for Ada:

```
DEFINE PLACEHOLDER LSE$BLOCK_COMMENT /TYPE=NOTERMINAL
    "- {tbs}"
    "-"
END DEFINE

DEFINE PLACEHOLDER LSE$LINE_COMMENT /TYPE=NOTERMINAL
    "- {tbs}"
END DEFINE

DEFINE PLACEHOLDER LSE$GENERIC /TYPE=NOTERMINAL
    "{tbs}"
END DEFINE
```

The following is an example definition for COBOL:

```
DEFINE PLACEHOLDER LSE$BLOCK_COMMENT
```

```
    "*"/INDENTATION=(fixed,1), "{tbs}"
    "*"/INDENTATION=(fixed,1)
```

The following is an example definition for Fortran:

```
DEFINE PLACEHOLDER LSE$BLOCK_COMMENT
    "!", " {tbs}"/INDENTATION=EXPAND
    "!"
```

ENTER COMMENT/BLOCK

Key	Keypad Mode
PF1-B	All

ENTER COMMENT/LINE

Key	Keypad Mode
PF1-L	All

Related Commands

UNDO ENTER COMMENT

Examples

The following are examples of converting pseudocode to comments:

1. `«This is something interesting.»`

   Entering the ENTER COMMENT/LINE command causes LSE to convert the pseudocode placeholder to a comment, as follows:

   ```
   {tbs} ! This is something interesting.
   ```

2. ```
   «We will move the third item from the left to be the»
   «next to the last item from the right in this case.»
   ```

   If there is a selected range active for both lines,entering the ENTER COMMENT/BLOCK command causes LSE to convert pseudocode to comments, as follows:

   ```
   - We will move the third item from the left to be the
   - next to the last item from the right in this case.{tbs}
   ```

ENTER LINE

ENTER LINE — Splits the current line into two lines.

Format

ENTER LINE

Qualifiers	Defaults
/BEGINNING	/BEGINNING
   ```

Qualifiers	Defaults
/[NO]COMMENT	/COMMENT
/END	/BEGINNING

Qualifiers

/BEGINNING (D)

Indicates that the cursor should be left at the beginning of the second line. If you position the cursor at the end of the original line, the /BEGINNING qualifier adds a new blank line to the current buffer and repositions the cursor at the beginning of the new line.

If you position the cursor at the beginning of a line, the/BEGINNING qualifier adds a new blank line before the current line and the cursor remains at the beginning of the current line.

If you position the cursor within a line, the /BEGINNING qualifier splits that line into two lines at the original cursor position and repositions the cursor at the beginning of the second line.

/COMMENT (D)
/NOCOMMENT

Indicates whether the second line should be a comment. This qualifier has no effect unless each of the following conditions are met:

- The current buffer is associated with a language.
- Comments are defined for the language.
- The cursor is positioned within a comment.
- Wrapping is set for the current buffer.

If all these conditions apply, you use the /NOCOMMENT qualifier when you want to terminate a comment and begin a code line.

/END

Indicates that the cursor should be left at the end of the first line. If you start with the cursor at the end of the original line, the/END qualifier causes the cursor to stay there.

If you start with the cursor at the beginning of a line, the /END qualifier adds a new blank line before the current line and positions the cursor on that blank line.

If you position the cursor within a line, specifying the /END qualifier splits the line in two leaving the cursor at the end of the first line.

Description

The ENTER LINE command splits the current line into two lines and places the cursor at the end of the first line or the beginning of the second line,depending on the qualifier you specify.

The ENTER LINE command also works in conjunction with the SET WRAP command to let you fill lines of text between margins. If wrapping is set for the buffer, LSE indents the second line to the left margin.

Keypad Equivalent

ENTER LINE/BEGINNING

Key	Keypad Mode
Return	All

ENTER LINE/END

Key	Keypad Mode
PF1-KP0 **OPEN LINE**	EDT LK201, EDT VT100, EVE LK201

ENTER LINE/NOCOMMENT

Key	Keypad Mode
PF1-Return	All

Related Commands

ENTER SPACE

SET WRAP

ENTER PSEUDOCODE

ENTER PSEUDOCODE — Inserts pseudocode placeholder delimiters.

Format

ENTER PSEUDOCODE

Description

The ENTER PSEUDOCODE command inserts pseudocode placeholder delimiters and positions the cursor on the first character of the right delimiter. The pseudocode placeholder delimiters must be defined before using this command.

If the cursor is on a placeholder defined with the command DEFINE PLACEHOLDER/PSEUDOCODE, the command has the usual effects of text insertion on the defined placeholders. The defined placeholder is autoerased and, if it is a list placeholder, it is duplicated.

If the cursor is on a placeholder defined with the command DEFINE PLACEHOLDER/NOPSEUDOCODE, or is on a pseudocode placeholder, the command is not allowed and a warning message is displayed.

If the cursor is not on a placeholder, the command inserts the pseudocode placeholder delimiter.

Keypad Equivalent

Key	Keypad Equivalent
PF1-Space bar	All

Related Commands

DEFINE LANGUAGE

DEFINE PLACEHOLDER

MODIFY LANGUAGE

Examples

The following are examples of entering pseudocode:

1. `IF {expression}`

   Entering the ENTER PSEUDOCODE command causes LSE to insert pseudocode placeholder delimiters, as follows:

   ```
   IF «»
   ```

2. `{statement} . . .`

   Entering the ENTER PSEUDOCODE command causes LSE to insert pseudocode placeholder delimiters, as follows:

   ```
   «»[statement] . . .
   ```

ENTER SPACE

ENTER SPACE — Inserts or overstrikes a space at the current cursor position,depending on whether the current editing mode is insert or overstrike. If wrap mode is set, line-oriented filling occurs.

Format

ENTER SPACE

Description

The ENTER SPACE command either inserts or overstrikes a space, depending on the current editing mode. If the cursor is past the right margin and wrap mode is set, the ENTER SPACE command performs a line-fill operation on the current line(see the SET [NO]WRAP command). You can change the right margin with the SET RIGHT_MARGIN command.

Keypad Equivalent

Key	Keypad Equivalent
Space bar	All

Related Commands

ENTER LINE

SET WRAP

ENTER SPECIAL

ENTER SPECIAL — Causes LSE to insert into the current buffer a character whose ASCII code you specify.

Format

ENTER SPECIAL ASCII-code

Parameter

ASCII-code

Specifies the ASCII code of the character you want as a decimal number from 0 through 255.

Description

The ENTER SPECIAL command inserts a special character into the buffer at the current cursor position. You can insert a form feed or other nonprinting characters as well as printing characters, such as letters and punctuation marks. When you enter the command, LSE prompts you for the ASCII code of the character you want to insert.

Keypad Equivalent

Table 2.4. ENTER SPECIAL

Key	Keypad Mode
PF1-KP3 **SPECINS**	EDT LK201, EDT VT100, EVE LK201
Ctrl/V	All

Related Commands

QUOTE

Examples

```
LSE> ENTER SPECIAL 12
```

Causes LSE to insert a form-feed character (Ctrl/L).

ENTER TAB

ENTER TAB — Inserts tabs and blanks at the current cursor position.

Format

ENTER TAB

Description

The ENTER TAB command inserts tabs and blanks at the current cursor position. If the cursor is at the beginning of the line, LSE inserts tabs and blanks up to the current indentation level. If the current indentation level is set at the beginning of the line, the ENTER TAB command does not insert tabs and blanks. If the cursor is not at the beginning of the line, the ENTER TAB command inserts an ASCII tab character.

Related Commands

SET INDENTATION

SET TAB_INCREMENT

TAB

ENTER TEXT

ENTER TEXT — Inserts text at the current cursor position.

Format

ENTER TEXT string

Parameter

string

Is a quoted string specifying the text to be inserted.

Description

The ENTER TEXT command inserts text from a quoted string at the current cursor position.

Example

```
LSE> ENTER TEXT "Insert this"
```

Inserts the quoted text Insert this at the current cursor position.

ERASE CHARACTER

ERASE CHARACTER — Erases a single character at the current cursor position.

Format

ERASE CHARACTER

Qualifiers	Defaults
/CURRENT	/CURRENT

Qualifiers	Defaults
/FORWARD	/CURRENT
/INDICATED	/INDICATED
/REVERSE	/CURRENT
/TO	/INDICATED

Qualifiers

/CURRENT (D)

Erases text in the current direction.

/FORWARD

Erases text in the forward direction.

/INDICATED (D)

Deletes the character at the current cursor position.

/REVERSE

Erases text in the reverse direction.

/TO

Deletes the character at the current cursor position when the direction is FORWARD. Deletes the character before the current cursor position when the direction is REVERSE.

Description

The ERASE CHARACTER command removes a single character from the current buffer. (A line terminator or ASCII tab character is considered one character.) In either insert or overstrike mode, the remainder of the line moves left one character to close up the space. An exception is the ERASE/TO CHARACTER/REVERSE command, which in overstrike mode changes the erased character to a space and moves left one position.

When the cursor is at the end of a line, the carriage return is deleted, and the text from the following line moves up to the right of the text in the current line.

Keypad Equivalent

ERASE/TO CHARACTER/REVERSE

Key	Keypad Mode
Delete	All

ERASE/TO CHARACTER/FORWARD

Key	Keypad Mode
Keypad comma (,) **DEL C**	EDT LK201, EDT VT100, EVE LK201

Key	Keypad Mode
None	EVE VT100

Related Commands

UNERASE CHARACTER

Example

```
LSE> ERASE CHARACTER
```

Deletes the character at the current cursor position (equivalent to pressing the comma key on the EDT numeric keypad).

ERASE LINE

ERASE LINE — Removes a line of text at the current cursor position.

Format

ERASE LINE

Qualifiers	Defaults
/BEGINNING	/BEGINNING
/CURRENT	/CURRENT
/END	/BEGINNING
/FORWARD	/CURRENT
/INDICATED	/INDICATED
/REVERSE	/CURRENT
/TO	/INDICATED

Qualifiers

/BEGINNING (D)

Indicates that the cursor should be moved to the beginning of a line as part of the ERASE operation. You cannot use the /BEGINNING qualifier with the /INDICATED qualifier.

/CURRENT (D)

Erases text in the current direction.

/END

Indicates that the cursor should be moved to the end of a line as part of the ERASE operation. You cannot use the /END qualifier with the /INDICATED qualifier.

/FORWARD

Erases text in the forward direction.

/INDICATED (D)

Erases the entire line that the cursor is on (including the carriage return and line feed), regardless of the cursor position within that line or the direction specified. The cursor moves to the beginning of the next line. You cannot use the /INDICATED qualifier with the/BEGINNING, /END, or /TO qualifiers.

/REVERSE

Erases text in the reverse direction.

/TO

Erases text from the current cursor position to the next line in the direction specified.

Description

The ERASE LINE command removes a line of text from the current cursor position. When LSE deletes all the text from the current cursor position to the end of the current line, the text on the following line moves up to fill the space to the right of the cursor.

Keypad Equivalent

ERASE/TO LINE/BEGINNING/REVERSE

Key	Keypad Mode
Ctrl/U	All

ERASE/TO LINE/BEGINNING/FORWARD

Key	Keypad Mode
PF4 **DEL L**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

ERASE/TO LINE/END

Key	Keypad Mode
PF1-KP2 **DEL EOL**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

Related Commands

`UNERASE LINE`

Examples

1. `LSE>` ***`ERASE LINE`***

Erases the entire line that the cursor is on, regardless of cursor position or direction specified.

2. **Ctrl/U**

   Erases text from the current cursor position to the beginning of the current line. If the cursor is already at the beginning of a line, Ctrl/U erases to the beginning of the previous line.

3. LSE> ***ERASE/TO LINE/END***

   Erases text from the current cursor position to the end of the current line, but does not erase the line break.

ERASE PLACEHOLDER

ERASE PLACEHOLDER — Deletes the text of a placeholder and related punctuation.

Format

ERASE PLACEHOLDER

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/[NO]GOTO_PLACEHOLDER	/GOTO_PLACEHOLDER
/REVERSE	/CURRENT

Qualifiers

/CURRENT (D)

Specifies cursor motion in the current direction.

/FORWARD

Specifies cursor motion in the forward direction.

/GOTO_PLACEHOLDER (D)
/NOGOTO_PLACEHOLDER

Specifies whether the cursor should move to the next placeholder after performing the ERASE operation. The movement to the next placeholder does not take place if it would force the current position to scroll off the screen.

/REVERSE

Specifies cursor motion in the reverse direction.

Description

The ERASE PLACEHOLDER command moves the cursor to the next placeholder in the direction specified and deletes the placeholder. The implicit GOTO PLACEHOLDER command caused by

the ERASE PLACEHOLDER command goes only to regular LSE placeholders, not to pseudocode placeholders. If the cursor is already on a placeholder, the deletion occurs in place.

If the cursor is on a character of a closing pseudocode placeholder delimiter, or not on a placeholder, the ERASE PLACEHOLDER command performs a GOTO PLACEHOLDER command before erasing.

If no placeholder is found, LSE returns an error message.

After deleting the placeholder and any leading tabs or blanks, LSE then deletes any leading separator text, or leading and trailing punctuation. If the resulting line or line segment is now empty, LSE then deletes the entire line or line segment.

Keypad Equivalent

ERASE PLACEHOLDER/FORWARD

Key	Keypad Mode
Ctrl/K	All

Related Commands

DEFINE PLACEHOLDER

UNERASE PLACEHOLDER

Examples

```
DEFINE PLACEHOLDER identifier_list -
    /TRAILING=":" -
    /SEPARATOR=","
               . . .
```

This DEFINE PLACEHOLDER specification applies to each of the following examples. The line comment delimiter is a double hyphen (--).

1. `<text> [identifier_list] <more text>`

   If this is the original text, entering an ERASE PLACEHOLDER command produces the following:

   `<text> <more text>`

2. `<text> [identifier_list] :  <more text>`

   If this is the original text, entering an ERASE PLACEHOLDER command produces the following:

   `<text> <more text>`

3. `<text> , [identifier_list] <more text>`

   If this is the original text, entering an ERASE PLACEHOLDER command produces the following:

   `<text> <more text>`

4. `<text> - [identifier_list] <more text>`

   If this is the original text, entering an ERASE PLACEHOLDER command produces the following:

```
<text> - <more text>
```

5. `- [identifier_list] <more text>`

If this is the original text, entering an ERASE PLACEHOLDER command produces the following:

```
- <more text>
```

ERASE SELECTION

ERASE SELECTION — Removes the text within the selected range.

Format

ERASE SELECTION

Description

The ERASE SELECTION command removes the text within the selected range. The selected range is the text between the select marker (see the SET SELECT_MARK command) and the current cursor position.

Related Commands

UNERASE SELECTION

Example

```
LSE>  ERASE SELECTION
```

Removes the text within the selected range.

ERASE WORD

ERASE WORD — Removes a word at the current cursor position.

Format

ERASE WORD

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/INDICATED	/INDICATED
/NEXT	
/PREVIOUS	
/REVERSE	/CURRENT

Qualifiers	Defaults
/TO	/INDICATED

Qualifiers

/CURRENT (D)

Erases text in the current direction.

/FORWARD

Erases text in the forward direction.

/INDICATED (D)

Deletes the entire word the cursor is on, regardless of the cursor's position within that word.

/NEXT

Erases the word following the cursor. When the cursor is positioned on a space, LSE erases all the spaces before and after the deleted word except one space. If the cursor is at the end of a line, the next line is appended to the current line. You cannot use the /NEXT qualifier with any other ERASE WORD qualifier.

/PREVIOUS

Erases the previous word when the cursor is on the first character of a word or between words. When the cursor is in the middle of a word, that entire word is erased and the cursor moves on to the first letter of the next word. You cannot use the /PREVIOUS qualifier with any other ERASE WORD qualifier.

/REVERSE

Erases text in the reverse direction.

/TO

Deletes text from the current cursor position to the beginning of the next word in the specified direction.

Description

The ERASE WORD command removes a word from the current buffer. A word can be terminated by tabs or characters not specified in the /IDENTIFIER_CHARACTERS qualifier on the DEFINE LANGUAGE command. A word can consist of identifier characters and trailing blanks, or it can consist of a single nonblank,nonidentifier character.

Keypad Equivalent

ERASE/TO WORD/REVERSE

Key	Keypad Mode
F13 **DEL PRV W**	EDT LK201
Ctrl/J **LINEFEED**	All

ERASE/TO WORD/FORWARD

Key	Keypad Mode
Keypad minus (–) **DEL W**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

ERASE WORD/NEXT

Key	Keypad Mode
Keypad comma (,) **ERASE WORD**	EVE VT100
F13 **ERASE WORD**	EVE LK201

Related Commands

UNERASE WORD

Example

```
LSE> ERASE WORD
```

Deletes the entire word at the current cursor position.

EXIT

EXIT — Ends an LSE editing session or SCA query session, and returns control to the calling process or the OpenVMS command language interpreter.

Format

EXIT

Qualifier	Defaults
/[NO]LOG {SCA only}	/NOLOG

Qualifier

/LOG
/NOLOG (D)

Indicates whether completion of an SCA session is reported.

Description

The EXIT command ends or suspends your session and returns control to the process that called LSE or SCA (usually the DCL command interpreter). If you are using LSE, the contents of buffers associated with files are written to their files if they have been modified. Buffers with the READ_ONLY attribute are not written back.

Keypad Equivalent

Key	Keypad Equivalent
F10 **EXIT**	EDT LK201, EVE LK201
None	EDT VT100, EVE VT100

DECwindows Interface Equivalent

Pull-down menu: File > Exit

Related Commands

ATTACH

QUIT

SPAWN

Examples

1. `LSE> EXIT`

   Ends an LSE session and writes modified buffers back to their respective files.

2. `SCA> EXIT`

   Ends an SCA query session.

EXPAND

EXPAND — Replaces placeholders, token names, alias names, or routine names at the current cursor position with the appropriate body of text or code,if the cursor is not on the overview line. Replaces the overview line with the underlying source lines if the cursor is on the overview line. Expands symbols to include their occurrences if the cursor is in a query buffer.

Format

EXPAND

Qualifiers	Defaults
/DEPTH= *n*	/DEPTH=1
/[NO]GOTO_PLACEHOLDER	/GOTO_PLACEHOLDER

Qualifiers

/DEPTH=n
/DEPTH=1 (D)

Specifies how many levels of detail are displayed. If you specify the value ALL, all subgroups for this overview line are expanded.

If the cursor is not on an overview line or is in a query buffer, the /DEPTH qualifier is ignored. Note that when you use the EXPAND command with SCA, this command does not support the /DEPTH qualifier.

/GOTO_PLACEHOLDER (D)
/NOGOTO_PLACEHOLDER

Specifies whether the cursor should move to the next placeholder after performing the EXPAND operation. The movement to the next placeholder does not take place if it would force the current position to scroll off the screen.

If the cursor is on an overview line, the /GOTO_PLACEHOLDER qualifier is ignored. Note that when you use the EXPAND command with SCA, this command does not support the /GOTO_PLACEHOLDER qualifier.

Description

If the cursor is not on an overview line,the EXPAND command expands text representing alias names, routine names,token names, or placeholders at the current position.

The EXPAND_CASE setting (defined with the DEFINE LANGUAGE or MODIFY LANGUAGE command) determines the case of the inserted text. If the EXPAND_CASE is UPPER or LOWER, LSE inserts the text in that case. If the EXPAND_CASE is AS_IS, LSE inserts the text as it appears in the token definition.

If the cursor is on an overview line, the overview is expanded to display the underlying hidden text.

The editor determines the relative level of detail of a line by comparing the indentation of the line with the indentation of other lines. The editor's treatment of the indentation of a line is influenced by indentation adjustment definitions. For more information,see the DEFINE ADJUSTMENT command.

For SCA, if the cursor is positioned on a symbol in a query buffer, the EXPAND command expands the symbol to display its occurrences.

Keypad Equivalent

EXPAND

Key	Keypad Mode
Ctrl/E	EDT LK201, EDT VT100
Ctrl//	EVE LK201, EVE VT100

EXPAND/DEPTH=ALL

Key	Keypad Mode
PF1- <	All

Related Commands

COLLAPSE

DEFINE ADJUSTMENT

DEFINE LANGUAGE/OVERVIEW_OPTIONS

FOCUS

MODIFY LANGUAGE

SET NOOVERVIEW

SET OVERVIEW

UNEXPAND

VIEW SOURCE

Examples

The following are examples of replacing a token or nonterminal placeholder with its body text based on the token or placeholder definition.

1.
```
DEFINE TOKEN for -
    /LANGUAGE=C -

    "for ([@expression@];  [@expression@];  [@expression@])"
    "{@statement@}"/INDENTATION=(EXPAND,1,TAB)
END DEFINE

DEFINE TOKEN "{" -
    /LANGUAGE=C -
    "{"/INDENTATION=PREVIOUS
    "{@statement@}..."/INDENTATION=(PREVIOUS, 1, TAB)
    "}"/INDENTATION=PREVIOUS
END DEFINE
```

   With the definitions in this example, typing "{" on the placeholder {@statement@} (Step 1) and expanding it (Step 2) produces the following(Step 3):

```
Step 1:
    for ( i = 0; i >15; i++ )
        {@statement@}

Step 2:
    for ( i = 0; i >15; i++ )
        {Step 3:    for ( i = 0; i >15; i++ )
    {
        {@statement@}...
    }
```

2.
```
DEFINE PLACEHOLDER "#IF" -
    /LANGUAGE= C -
    "#if {@constant expression@}"/INDENTATION=(FIXED,1)
    "[@#else_clause@]"/INDENTATION=(FIXED,1)
    "#endif"/INDENTATION=(FIXED,1)
END DEFINE
```

   With the definitions in this example, expanding the [@#IF@] placeholder at any column always yields indentation to the column defined, as follows:

```
Step1:
        [@#if@]

Step2:
#if {@constant expression@}[@
#else_clause@]
#endif
```

EXTEND

EXTEND — Compiles one or more DECTPU procedures to extend LSE.

Format

EXTEND $\left\{\begin{matrix}\text{Procedure-name}\\ *\end{matrix}\right\}$

Qualifier	Defaults
/INDICATED	/INDICATED

Qualifier

/INDICATED (D)

If you specify the /INDICATED qualifier, the EXTEND command compiles the DECTPU procedure in which the cursor is located. You cannot specify the /INDICATED qualifier with a parameter.

Parameters

procedure-name

The name of the DECTPU procedure you want to compile. You can abbreviate the procedure name.

*****

Wildcard symbol instructing DECTPU to compile all the procedures and statements in the buffer.

Description

The EXTEND command compiles one or more DECTPU procedures to extend LSE. Using EXTEND without specifying the procedure name compiles the procedure in which the cursor is located.

To execute a compiled procedure, use the EXTEND command followed by the name of the procedure you want executed. To save a compiled procedure in a section file for future editing sessions, use the SAVE SECTION command.

If the procedure contains any overview records, a message informs you that the operation cannot be performed because there are overview records in the selected range. Compiler messages appear in the message window.

You cannot specify a parameter with the /INDICATED qualifier.

Example

```
LSE> EXTEND user_proc
```

Compiles a procedure called USER_PROC.

EXTRACT ADJUSTMENT

EXTRACT ADJUSTMENT — Extracts the definition of the named adjustment and formats the definition as a command.

Format

EXTRACT ADJUSTMENT adjustment-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the adjustment being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the adjustment definitions defined during this editing session should be extracted.

Parameter

adjustment-name

Specifies the name of the adjustment you want. You can specify a wildcard.

Description

The EXTRACT ADJUSTMENT command extracts the specified adjustment definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE ADJUSTMENT command precedes the corresponding DEFINE ADJUSTMENT command.

With the EXTRACT ADJUSTMENT command, you can modify definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can use this command to extract adjustments to make global changes to them. After you edit the buffer, use the DO command to execute the changes.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE ADJUSTMENT

DELETE ADJUSTMENT

SHOW ADJUSTMENT

Examples

```
LSE> EXTRACT ADJUSTMENT/LANGUAGE=ADA then
```

Extracts the current definition of the then adjustment from the list of adjustments associated with the Ada language and places the definition at the end of the current buffer.

EXTRACT ALIAS

EXTRACT ALIAS — Extracts the definition of an alias and formats the definition as a command.

Format

EXTRACT ALIAS alias-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the alias being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the definitions of aliases defined during this editing session should be extracted.

Parameter

alias-name

Specifies the name of the alias you want. You can specify a wildcard.

Description

The EXTRACT ALIAS command extracts the specified alias definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be

read back and replace existing definitions. Specifically, the DELETE ALIAS command precedes the corresponding DEFINE ALIAS command.

With the EXTRACT ALIAS command, you can modify alias definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE ALIAS

DELETE ALIAS

SHOW ALIAS

Example

```
LSE> EXTRACT ALIAS EXE
```

Places the current definition of the EXE alias at the end of the current buffer.

EXTRACT KEYWORDS

EXTRACT KEYWORDS — Extracts the definition of the specified keyword list and formats the definition as a command.

Format

EXTRACT KEYWORDS keyword-list-name

Qualifier	Defaults
/NEW	

Qualifier

/NEW

Specifies that only the definitions of keyword list names defined during this editing session should be extracted.

Parameter

keyword-list-name

Specifies the keyword list name. You can specify a wildcard.

Description

The EXTRACT KEYWORDS command extracts the specified keyword list definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits

them to be read back and replace existing definitions. Specifically, the DELETE KEYWORDS command precedes the corresponding DEFINE KEYWORDS command.

With the EXTRACT KEYWORDS command, you can modify keyword-list definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE KEYWORDS

DELETE KEYWORDS

SHOW KEYWORDS

Example

```
LSE> EXTRACT KEYWORDS author_name
```

Places the current definition of the keyword list *author_name* at the end of the current buffer.

EXTRACT LANGUAGE

EXTRACT LANGUAGE — Extracts the definition of the specified language and formats the definition as a command.

Format

EXTRACT LANGUAGE language-name

Qualifier	Defaults
/NEW	

Qualifier

/NEW

Specifies that only the definitions of languages defined during this editingsession should be extracted.

Parameter

language-name

Specifies the name of the language you want. You can specify a wildcard.

Description

The EXTRACT LANGUAGE command extracts the specified language definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits

them to be read back and replace existing definitions. Specifically, the DELETE LANGUAGE command precedes the corresponding DEFINE LANGUAGE commands.

With the EXTRACT LANGUAGE command, you can modify language definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE LANGUAGE

DELETE LANGUAGE

SHOW LANGUAGE

Example

```
LSE> EXTRACT LANGUAGE Pascal
```

Places the current definition of the Pascal language at the end of the current buffer.

EXTRACT MODULE

EXTRACT MODULE — Extracts specified modules of source-analysis data from an SCA library.

Format

EXTRACT MODULE module-name-expr[, . . .

Qualifiers	Default
/DECLARATION_CLASS=declaration-class	
/LIBRARY=library-spec	
/[NO]LOG	/NOLOG
/OUTPUT=file-spec	

Qualifiers

/DECLARATION_CLASS=declaration-class

Indicates the class of the module to be copied. The following declaration classes are supported:

- PRIMARY—Module implementation
- ASSOCIATED—Module specification

If you do not specify a declaration class, SCA extracts both classes, if they exist.

/LIBRARY=library-spec

Specifies the SCA static library from which to extract the module. This library must be one of the current SCA libraries (established by a SET LIBRARY command).If you do not specify this

qualifier, SCA tries to extract the module from the primary library (the first of the current SCA libraries).

/LOG
/NOLOG (D)

Indicates whether SCA reports the extraction of a module.

/OUTPUT=file-spec

Specifies the file into which all modules of source-analysis data will be written. The default is /OUTPUT=module-name.ANA, where the module name is the name of the file the compiler created.

Parameter

module-name-expr[, . . .]

Specifies the modules to extract. If you specify more than one library, SCA extracts the module from the first library in which it occurs.

Description

The EXTRACT MODULE command extracts the specified module from the specified SCA static library and places it in a file of type .ANA, which is the file type for source-analysis data files created by compilers. The EXTRACT MODULE command performs the reverse function of the LOAD command.

Related Commands

LOAD

SET LIBRARY

Example

```
$ SCA EXTRACT MODULE module_1
```

Extracts module_1 from the current library.

EXTRACT PACKAGE

EXTRACT PACKAGE — Extracts the definition of the specified package and formats the definition as a command.

Format

EXTRACT PACKAGE package-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the package being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the definitions of packages defined during this editing session should be extracted.

Parameter

package-name

Specifies the name of the package you want. You can specify a wildcard.

Description

The EXTRACT PACKAGE command extracts the specified package definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE PACKAGE command precedes the corresponding DEFINE PACKAGE command.

With the EXTRACT PACKAGE command, you can modify package definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE PACKAGE

DELETE PACKAGE

SHOW PACKAGE

Example

```
LSE> EXTRACT PACKAGE system_services
```

Places the current definition of the *system_services* package at the end of the current buffer.

EXTRACT PARAMETER

EXTRACT PARAMETER — Extracts the definition of the specified parameter and formats the definition as a command.

Format

EXTRACT PARAMETER parameter-name

Qualifiers	
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the parameter being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the definitions of parameters defined during this editing session should be extracted.

Parameter

parameter-name

Specifies the name of the parameter you want. You can specify a wildcard.

Description

The EXTRACT PARAMETER command extracts the specified parameter definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE PARAMETER command precedes the corresponding DEFINE PARAMETER command.

With the EXTRACT PARAMETER command, you can modify definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE PARAMETER

DELETE PARAMETER

SHOW PARAMETER

Example

```
LSE> EXTRACT PARAMETER id
```

Places the current definition of the *id* parameter at the end of the current buffer.

EXTRACT PLACEHOLDER

EXTRACT PLACEHOLDER — Extracts the definition of the specified placeholder and formats the definition as a command.

Format

EXTRACT PLACEHOLDER placeholder-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the placeholder being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the placeholder definitions defined during this editing session should be extracted.

Parameter

placeholder-name

Specifies the name of the placeholder you want. You can specify a wildcard.

Description

The EXTRACT PLACEHOLDER command extracts the specified placeholder definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE PLACEHOLDER command precedes the corresponding DEFINE PLACEHOLDER command.

With the EXTRACT PLACEHOLDER command, you can modify definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can use this command to extract placeholders to make global changes to them,such as changing delimiters or placeholder names. Use the SET NOAUTO_ERASE command to avoid erasing the placeholders as you type within their delimiters and perform other edits. After you edit the buffer, use the DO command to execute the changes.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE PLACEHOLDER

DELETE PLACEHOLDER

SHOW PLACEHOLDER

Example

```
LSE> EXTRACT PLACEHOLDER/LANGUAGE=ADA text
```

Extracts the current definition of the text placeholder from the list of placeholders associated with the Ada language and places the definition at the end of the current buffer.

EXTRACT ROUTINE

EXTRACT ROUTINE — Extracts the definition of the specified routine and formats the definition as a command.

Format

EXTRACT ROUTINE routine-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the routine being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the definitions of routines defined during this editing session should be extracted.

Parameter

routine-name

Specifies the name of the routine you want. You can specify a wildcard.

Description

The EXTRACT ROUTINE command extracts the specified routine definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE ROUTINE command precedes the corresponding DEFINE ROUTINE command.

With the EXTRACT ROUTINE command, you can modify definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE ROUTINE

DELETE ROUTINE

SHOW ROUTINE

Example

```
LSE> EXTRACT ROUTINE add_holder
```

Places the current definition of the *add_holder* routine at the end of the current buffer.

EXTRACT TAG

EXTRACT TAG — Extracts the definition of the specified tag and formats the definition as a command.

Format

EXTRACT TAG tag-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the tag being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the tag definitions defined during this editing session should be extracted.

Parameter

tag-name

Specifies the name of the tag you want. You can specify a wildcard.

Description

The EXTRACT TAG command extracts the specified tag definition and formats it as a command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE TAG command precedes the corresponding DEFINE TAG command.

With the EXTRACT TAG command, you can modify definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE TAG

DELETE TAG

SHOW TAG

Example

```
LSE> EXTRACT TAG/LANGUAGE=ADA text
```

Extracts the current definition of the text tag from the list of tags associated with the Ada language and places the definition at the end of the current buffer.

EXTRACT TOKEN

EXTRACT TOKEN — Extracts the definition of the specified token and formats the definition as a command.

Format

EXTRACT TOKEN token-name

Qualifiers	Defaults
/LANGUAGE[=language-name]	
/NEW	

Qualifiers

/LANGUAGE[=language-name]

Specifies the language associated with the token being extracted. If you do not specify the /LANGUAGE qualifier, the default is the current language.

/NEW

Specifies that only the definitions of tokens defined during this editing session should be extracted.

Parameter

token-name

Specifies the name of the token you want. You can specify a wildcard.

Description

The EXTRACT TOKEN command extracts the specified token definition and formats it asa command. LSE inserts the specified definitions at the end of the current buffer in a form that permits them to be read back and replace existing definitions. Specifically, the DELETE TOKEN command precedes the corresponding DEFINE TOKEN command.

With the EXTRACT TOKEN command, you can modify definitions by editing and then executing them using the DO command. You can write definitions to a file.

You can use this command to extract tokens to make global changes to them,such as changing delimiters or token names. Use the SET NOAUTO_ERASE command to avoid erasing the tokens as you type within their delimiters and perform other edits. After you edit the buffer, use the DO command to execute the changes.

You can create new definitions in a buffer, and edit and execute them until they are correct.

Related Commands

DEFINE TOKEN

DELETE TOKEN

SHOW TOKEN

Example

```
LSE> EXTRACT TOKEN WHILE
```

Places the current definition of the WHILE statement at the end of the current buffer.

For additional examples, see the section about redefining language elements in the *Guide to Language-Sensitive Editor for OpenVMS Systems*.

FILL

FILL — Reformats the text within a selected range to put as much text on a line as possible. This command is particularly useful for comments and ordinary prose, but is not normally used with program code.

Format

FILL

Qualifiers	Defaults
/COMMENT_COLUMN=	/COMMENT_COLUMN=
——CONTEXT_DEPENDENT	——CONTEXT_DEPENDENT
/COMMENT_COLUMN=number	/COMMENT_COLUMN=
	——CONTEXT_DEPENDENT

Qualifiers

/COMMENT_COLUMN=CONTEXT_DEPENDENT (D)
/COMMENT_COLUMN=number

Specifies that the comment column should be determined from the context. LSE uses the position of the commented segment in the first line of the selected range as the comment column.

The *number* specifies an explicit column number in which to align the comments. LSE aligns all commented segments in the selected range with this column; all paragraphs within the range have

the same comment-column setting. The *number* must be an integer in the range of from 1 to 131. The value must be consistent with the lengths of the comment delimiters used within the range.

For a text fill, LSE ignores this qualifier.

Description

The FILL command reformats the text in the selected range. The selected range is the text between the select marker (see the SET SELECT_MARK command) and the current cursor position. If you do not provide a selected range, the FILL command reformats the current paragraph. (Note that the current paragraph includes the text on all previous and subsequent lines until LSE encounters a completely blank line.) LSE preserves any blank lines you insert in the text.

If the buffer is associated with a language, and comment delimiters have been defined for the language, LSE just reformats the commented segments of the lines in the selected range. If the buffer is not associated with a language,or there are no comment delimiters, LSE performs a text fill.

The FILL command reformats a block of text so as many complete words as possible fit on each line without exceeding the right margin. You can change the right margin with the SET RIGHT_MARGIN command. Except in comments,the FILL command indents the reformatted text to the LEFT_MARGIN setting.

When you enter the FILL command, LSE treats spaces, tabs, and carriage returns as word delimiters. LSE treats character sequences as whole words if it recognizes such sequences as placeholders.

Keypad Equivalent

Key	Keypad Equivalent
PF1-KP8 **FILL**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

DECwindows Interface Equivalent

Pull-down menu: Edit > Fill

Related Commands

DEFINE LANGUAGE

SET SELECT_MARK

SET WRAP

Examples

The /COMMENT_COLUMN=CONTEXT_DEPENDENT qualifier (the default) is in effect in the following examples.

1.
```
IF (col >= R_Margin) THEN ! This is the start of an
    BEGIN                    ! extended end-of-line comment block
    i := i + 1 ;
```

```
      j := j + i ;  ! another comment
    !to be filled
```

Entering the FILL command for this example of line comments produces the following format:

```
IF (col >= R_Margin) THEN ! This is the start of an extended
    BEGIN                 ! end-of-line comment block
    i := i + 1 ;
    j := j + i ;          ! another comment to be filled
```

Note that the first word after the start of the comment on the second line (the word extended) was used to fill out the first line.

2.
```
IF (col >= R_Margin) THEN   (* This is the start of a *)
        BEGIN                (* bracketed comment sequence that *)
        VAR x: INTEGER;
  (* extends over several lines  *)
```

Entering the FILL command for this example of consecutive, single-line bracketed comments produces the following format:

```
IF (col >= R_Margin) THEN (* This is the start of a bracketed *)
        BEGIN             (* comment sequence that extends    *)
        VAR x: INTEGER;   (* over several lines               *)
```

FIND

FIND — Locates occurrences described by the current SCA libraries.

Format

`FIND query-expression`

Qualifiers	Defaults
/DESCRIPTION=string	
/[NO]DISPLAY[=(option, …)]	/DISPLAY=DEFAULT
/[NO]LOG	/LOG
/[NO]MODIFY[=query-name]	/NOMODIFY
/NAME=[query-name]	
/OUTPUT[=file-spec]	
/[NO]REPLACE	/NOREPLACE
/[NO]RESULT=option	/RESULT=DEFAULT
/[NO]SYNCHRONIZE	/NOSYNCHRONIZE

Qualifiers

/DESCRIPTION=string

Specifies a single line of text displayed along with the query name when the query is displayed by entering the SHOW QUERY command.

/DISPLAY[=(option, …)]
/DISPLAY=DEFAULT (D)
/NODISPLAY

Indicates how much information SCA displays about query results. Use one or more of the following keywords to request specific information:

Keyword	Description
NAME	Symbol name
CLASS	Class of item
LINE_NUMBER	Compilation line number
MODULE	Module name containing a symbol occurrence
FILE_SPEC	File name and type containing a symbol occurrence
FULL_FILE_SPEC	Complete file specification containing a symbol occurrence
RECORD_NUMBER	Record number within a source file
RELATIONSHIP	Relationship type
ROUTINE_NAME	Routine name containing a symbol occurrence
NUMBER	Number of the display line
OCCURRENCE_TYPE	Type of symbol occurrence (such as declaration, read, or call)
ALL	All of the previous options
DEFAULT	Default settings of the display options
NONE	Nothing (equivalent to the /NODISPLAY qualifier)

You can prefix any keyword (except ALL, DEFAULT, and NONE) with NO to request that the information be excluded.

The initial default for each type of new query is as follows:

```
DISPLAY=(NAME,CLASS,MODULE,LINE_NUMBER,OCCURRENCE_TYPE,RELATIONSHIP)
```

/LOG (D)
/NOLOG

Indicates whether the count of symbol occurrences will be reported.

/MODIFY[=query-name]
/NOMODIFY (D)

Indicates that an existing query is to be modified. By default,each FIND command creates a new query.

The /MODIFY=query-name qualifier indicates that the specified query should be modified according to the specification of the FIND command. The specified query must already exist.

By default, the /MODIFY qualifier specifies the current query.

/NAME[=query-name]

Specifies the name of the query. If a query with the same name already exists, you must also specify the /REPLACE qualifier. If a query name is not specified, then SCA assigns a unique name to the query. The query name can be a quoted string.

/OUTPUT[=file-spec]

Specifies that command output is to go to a file rather than be displayed on your screen (or go to a batch log file). The default output-file specification is SCA.LIS.

/REPLACE
/NOREPLACE (D)

Indicates whether existing queries should be replaced by new queries. By default, a FIND command that creates a query with the same name as an already existing query will fail.

/RESULT=option
/RESULT=DEFAULT (D)
/NORESULT

Indicates the type of query results displayed. You must specify one of the following keywords:

Keyword	Description
SYMBOLS	Only symbols are displayed.
OCCURRENCES	Symbols and occurrences are displayed.
DEFAULT	Either symbols or occurrences, or both, are displayed. SCA chooses the result type that is most appropriate for the current query.

The /NORESULT qualifier specifies that no results should be displayed. This means that no query evaluation is done. If a query result exists because you entered a FIND command,specifying /NORESULT causes that result to be deleted.

/SYNCHRONIZE
/NOSYNCHRONIZE (D)

Indicates that the query result must be synchronized with the current state of the virtual library being queried. By default,/NOSYNCHRONIZE causes SCA to do as little processing as necessary to evaluate the query. This can lead to query results that reflect the state of the virtual library at the time of a previous query.

The /SYNCHRONIZE qualifier specifies that the query result must be synchronized with the current virtual library. SCA attempts to minimize the amount of processing, but the result is still synchronized with the virtual library that was in effect at the time the query was evaluated.

Parameter

query-expression

Specifies the set of occurrences to be found.

For information on query expressions, see the chapters on query expressions and query language in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

Description

The FIND command locates occurrences described by the current SCA libraries. By default, each time you enter a FIND command, SCA creates a new query to describe the result. To remove queries you no longer need, use the DELETE QUERY command.

For more information about the FIND command, see the chapter on performing SCA tasks in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

DECwindows Interface Equivalent

FIND SYMBOL

Pull-down menu: Navigate > Find Symbol

Related Commands

COLLAPSE

DELETE QUERY

EXPAND

GOTO QUERY

GOTO SOURCE

NEXT QUERY

NEXT STEP

PREVIOUS QUERY

PREVIOUS STEP

SAVE QUERY

Examples

1. LSE> ***FIND build****

   Finds all occurrences of symbols whose name begins with *build*.

2. LSE> ***FIND/RESULT=SYMBOL copy_file and symbol=literal***

   Finds all occurrences of literals named *copy_file*. Only symbol information is included in the display.

3. LSE> ***FIND/RESULT=OCCURRENCE occ=primary and symbol=routine***

   Finds the primary declarations of all routines. Both symbol and occurrence information are included in the display.

4. LSE> ***FIND calling expand_string***

   Finds the routines that are calling *expand_string*.

5. `LSE> FIND called_by( translit, depth=all )`

   Displays the complete call-tree below *translit*.

6. `LSE> FIND typed_by( integer, symbol=variable )`

   Finds all the variables of type *integer*.

FOCUS

FOCUS — Displays an overview of the buffer. The current line remains visible, and the rest of the buffer is compressed.

Format

FOCUS

Description

The FOCUS command displays the current line and its surrounding text. The rest of the lines in the buffer are collapsed as much as possible and are represented by overview lines.

The editor determines the relative level of detail of a line by comparing the indentation of the line with the indentation of other lines. The editor's treatment of the indentation of a line is influenced by indentation adjustment definitions. For more information,see the DEFINE ADJUSTMENT command.

Keypad Equivalent

Key	Keypad Equivalent
PF1-period	All

DECwindows Interface Equivalent

COLLAPSE

Pop-up menu: Query buffer → Collapse

Pull-down menu: View → Collapse

COLLAPSE/DEPTH=ALL

Pull-down menu: View → Collapse All

Related Commands

DEFINE ADJUSTMENT

DEFINE LANGUAGE/OVERVIEW_OPTIONS

EXPAND

FOCUS

MODIFY LANGUAGE

SET NOOVERVIEW

SET OVERVIEW

VIEW SOURCE

GOTO BOTTOM

GOTO BOTTOM — Moves the cursor to the bottom of the current buffer.

Format

GOTO BOTTOM

Description

The GOTO BOTTOM command moves the cursor to the bottom of the current buffer. To achieve the same result, DECwindows interface users can use MB1 to drag the vertical scroll bar slider to the bottom of the scroll bar.

Keypad Equivalent

Key	Keypad Equivalent
PF1-KP4 **BOTTOM**	EDT LK201, EDT VT100, EVE LK201
PF1-E6	EDT LK201
PF1-↓	EVE LK201, EVE VT100

DECwindows Interface Equivalent

Pull-down menu: Navigate → Goto Bottom

Related Commands

GOTO TOP

GOTO BUFFER

GOTO BUFFER — Moves the cursor to the specified buffer.

Format

GOTO BUFFER buffer-name

Qualifiers	Defaults
/[NO]CREATE	/NOCREATE
/[NO]READ_ONLY	/READ_ONLY
/[NO]WRITE	/NOWRITE

Qualifiers

/CREATE
/NOCREATE (D)

Specifies whether the buffer should be created if it does not exist.

/READ_ONLY (D)
/NOREAD_ONLY

Specifies whether the specified buffer should have the read-only attribute. If the buffer has this attribute, LSE does not write the contents to a file when you exit from LSE, or when you enter a COMPILE command. This qualifier has an effect only if the GOTO BUFFER command is creating a buffer. If you are going to an already existing buffer, the read-write status of that buffer is not changed. The /WRITE qualifier is equivalent to the/NOREAD_ONLY qualifier.

/WRITE
/NOWRITE (D)

Specifies whether the specified buffer should have the write attribute. If the buffer has this attribute, LSE writes the contents of the buffer to a file when you exit from LSE, or when you enter a COMPILE command. This qualifier has an effect only if the GOTO BUFFER command is creating a buffer. If you are going to an already existing buffer, the read-write status of the buffer is not changed. The /NOREAD_ONLY qualifier is equivalent to the /WRITE qualifier.

Parameter

buffer-name

Specifies the name of the buffer. You can use abbreviations.

You can specify a buffer name with a character string value of up to 255alphanumeric or special characters. If you begin the buffer name with special characters, such as those accessed on the top row of your keyboard by pressing the shift key, you must enclose the buffer name in quotation marks. Similarly,to specify a name that contains embedded blanks (spaces), or quotation marks and spaces, enclose the entire string in quotation marks.

Description

The GOTO BUFFER command moves the cursor to the specified buffer. LSE maps the buffer to the current window, and moves the cursor to the last remembered position in that buffer.

You can use the mouse to select a buffer from the list displayed by the SHOW BUFFER command.

DECwindows Interface Equivalent

Pull-down menu: Source Goto → Buffer

Related Commands

GOTO FILE

NEXT BUFFER

PREVIOUS BUFFER

SHOW BUFFER

Example

```
LSE> GOTO BUFFER $SHOW
```

Causes LSE to display the buffer that contains the latest response to a SHOW command.

GOTO CHARACTER

GOTO CHARACTER — Moves the cursor to the next character.

Format

GOTO CHARACTER

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/HORIZONTALLY	/HORIZONTALLY
/REVERSE	/CURRENT
/VERTICALLY	/HORIZONTALLY

Qualifiers

/CURRENT (D)

Instructs LSE to use the current direction of the buffer.

/FORWARD

Instructs LSE to move the cursor down, or to the right.

/HORIZONTALLY (D)

Instructs LSE to move the cursor horizontally.

/REVERSE

Instructs LSE to move the cursor up, or to the left.

/VERTICALLY

Instructs LSE to move the cursor vertically.

Description

The GOTO CHARACTER command moves the cursor one character in the specified direction. LSE does not position the cursor when the screen is empty,unless text spaces have been created using the space bar. The cursor moves across tab characters and wraps at the edge of the screen.

You can use the mouse cursor to position the editing cursor to any text in an editing window.

Keypad Equivalent

GOTO CHARACTER/VERTICALLY/FORWARD

Key	Keypad Mode
Down ↓	All
KP2 ↓	EVE VT100

GOTO CHARACTER/HORIZONTALLY/REVERSE

Key	Keypad Mode
Left ←	All
KP1 ←	EVE VT100

GOTO CHARACTER/HORIZONTALLY/FORWARD

Key	Keypad Mode
Right →	All
KP3 →	EVE VT100

GOTO CHARACTER/VERTICALLY/REVERSE

Key	Keypad Mode
Up ↑	All
KP5 ↑	EVE VT100

GOTO CHARACTER/HORIZONTALLY/CURRENT

Key	Keypad Mode
KP3 **CHAR**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

Related Commands

GOTO LINE

GOTO WORD

GOTO COMMAND

GOTO COMMAND — Produces the LSE Command> prompt at which you can enter LSE or SCA commands.

Format

GOTO COMMAND

Description

The GOTO COMMAND command moves the cursor to the command region. With the DECwindows interface, you can use the mouse to move the cursor to the commands region.

Keypad Equivalent

Key	Keypad Equivalent
Do **DO**	EDT LK201, EVE LK201
PF1-KP7s	All

Related Commands

DO

GOTO DECLARATION

GOTO DECLARATION — Displays the declaration of the symbol specified. LSE displays the source code containing the symbol declaration in another window and positions the cursor on the symbol declaration.

Format

GOTO DECLARATION [symbol-name]

Qualifiers	Defaults
/ASSOCIATED	/PRIMARY
/CONTEXT_DEPENDENT	/PRIMARY
/INDICATED	
/PRIMARY	/PRIMARY

Qualifiers

/ASSOCIATED

Indicates that you want to see the associated declaration for the symbol. An associated declaration is a related declaration that accompanies the primary declaration (such as an EXTERNAL declaration).

/CONTEXT_DEPENDENT

If you specify both the /CONTEXT_DEPENDENT and the /INDICATED qualifiers, SCA determines which declaration to display using the following criteria:

- If the specified occurrence of the symbol is a reference, LSE displays the declaration specified by the compiler as bound to that occurrence of the symbol.
- If the specified occurrence of the symbol is an associated declaration, LSE displays the primary declaration.

- If the specified occurrence of the symbol is a primary declaration, LSE displays the associated declaration.

If you specify the /CONTEXT_DEPENDENT qualifier but not the /INDICATED qualifier, LSE displays the primary declaration.

/INDICATED

Instructs LSE to use the symbol name at the current cursor position, or the text within the currently active selected range, as the symbol name. To help SCA identify exactly which occurrence of the symbol name the cursor is positioned on, LSE passes both the current cursor position in the buffer and the file specification for the current buffer to SCA.

If SCA has no information for the symbol name at the current cursor position (for example, if the line containing the symbol is a new line and the file has not been recompiled), SCA uses whatever general information it has about that symbol, as if you entered a GOTO DECLARATION command for the symbol name without the /INDICATED qualifier.

If you specify the /INDICATED qualifier, you must not specify the *symbol-name* parameter.

/PRIMARY (D)

Indicates that you want to see the primary declaration for the symbol. A primary declaration is the declaration that SCA interprets as most significant for a symbol (such as a FUNCTION declaration). For example, the primary declaration of a routine describes the body of theroutine.

Parameter

symbol-name

Specifies that the declaration associated with the specified symbol is to be displayed. You must not specify a symbol name if you specify the /INDICATED qualifier.

Description

The GOTO DECLARATION command causes LSE to display the source for the declaration of the specified or indicated symbol.

If more than one declaration is to be displayed, LSE creates a new query to list those declarations.

Keypad Equivalent

GOTO DECLARATION/INDICATED/PRIMARY

Key	Keypad Mode
Ctrl/D	All

GOTO DECLARATION/INDICATED/CONTEXT_DEPENDENT

Key	Keypad Mode
PF1-Ctrl/D	All

DECwindows Interface Equivalent

GOTO DECLARATION/INDICATED

Pop-up menu: User buffer → Find Declaration

Related Commands

FIND

GOTO QUERY

GOTO SOURCE

Example

```
LOCAL   X;
    .
    .
    .
X = Y;
```

```
LSE> GOTO DECLARATION/INDICATED
```

Causes LSE to display the declaration LOCAL X if your cursor is positioned on the X of the assignment statement X = Y.

GOTO FILE

GOTO FILE — Moves the cursor to the buffer containing the specified file. If no buffer contains the specified file, LSE reads the file into a new buffer.

Format

GOTO FILE file-spec

Qualifiers	Default
/[NO]CREATE	/NOCREATE
/[NO]MODIFY	
/NEW	
/READ_ONLY	
/WRITE	

Qualifiers

/CREATE
/NOCREATE (D)

Specifies whether the GOTO FILE command should succeed if the specified file does not exist. This qualifier has no effect if you are going to an existing buffer.

/MODIFY
/NOMODIFY

Specifies whether the buffer you create is modifiable or unmodifiable. If you specify the /MODIFY qualifier, the GOTO FILE command creates a modifiable buffer. If you specify the /NOMODIFY qualifier, the GOTO FILE command creates an unmodifiable buffer. If you do not specify either qualifier, LSE determines the buffer's modifiable status from the read-only or write setting. By default, a read-only buffer is unmodifiable and a write buffer is modifiable.

/NEW

Specifies that you want to create a new file. If the specified file already exists, LSE reports an error and aborts the command. The *file-spec* parameter cannot contain wildcards if you specify this qualifier. You cannot use this qualifier with the /[NO]CREATE or/[NO]MODIFY qualifiers.

/READ_ONLY

Specifies that the buffer you create is read-only and therefore unmodifiable. This qualifier and the /WRITE qualifier override any setting established by the SET DIRECTORY command.

If you specify neither the /READ_ONLY nor the /WRITE qualifier, LSE uses the default established by the most recent SET DIRECTORY command for the directory that contains the file. If during your current editing session you have not entered a SET DIRECTORY command nor defined the logical LSE$READ_ONLY_DIRECTORY,the buffer is writable by default.

/WRITE

Specifies that the buffer you create is writable and therefore modifiable. This qualifier and the /READ_ONLY qualifier override any setting established by the SET DIRECTORY command.

If you specify neither the /WRITE nor the /READ_ONLY qualifier, LSE uses the default established by the most recent SET DIRECTORY command for the directory that contains the file. If during your current editing session you have not entered a SET DIRECTORY command nor defined the logical LSE$READ_ONLY_DIRECTORY,the buffer is writable by default.

Parameter

file-spec

Specifies the name of the file to be edited. LSE uses the directories specified in the SET SOURCE_DIRECTORY command to resolve the file specification. If the file cannot be found in one of those directories (or the list of directories is empty) and you used the /CREATE qualifier, LSE creates the file in your default directory.

Description

The GOTO FILE command moves the cursor to its last position in the buffer containing the specified file, if a buffer corresponding to the specified file already exists.

If no such buffer exists, LSE creates a new one, taking the buffer name from the name and type of the *file-spec* parameter. If that name is not unique, LSE prompts you for a buffer name and gives you the option of replacing an already existing buffer of the same name or canceling the command. If you do not cancel the command, LSE reads the specified file into the buffer, positions the cursor in that buffer, and maps the buffer to the current window.

If you do not specify either the /READ_ONLY or the /WRITE qualifier on the command, LSE sets the read and write status of the buffer based on the status of the directory in which the file is found. If the directory is a read-only directory (that is, if it is on the list established by the SET DIRECTORY/READ_ONLY command), LSE creates the buffer as read-only and unmodifiable; otherwise, the buffer is set writable and modifiable.

If the specified file is to be read in (that is, it is not already in a buffer), LSE uses CMS to fetch a copy of the file and place it in an unmodifiable buffer, if the directory for the file to be accessed is the same as your current CMS library. The GOTO FILE command uses the setting of the SET CMS command when performing a FETCH operation.

Note that you cannot use the GOTO FILE command to reserve files from your current CMS library. To reserve a file, use the RESERVE command.

DECwindows Interface Equivalent

GOTO FILE/NEW

Pull-down menu: File → New File . . .

GOTO FILE

Pull-down menu: File → Open File . . .

Related Commands

GOTO BUFFER

READ

SET CMS

SET DIRECTORY

Example

```
LSE> GOTO FILE x.y
```

Brings the file x.y into the current buffer.

GOTO LINE

GOTO LINE — Moves the cursor to the end of the line, or to the next line if the cursor is already at the end of a line.

Format

GOTO LINE

Qualifiers	Defaults
/BEGINNING	/BEGINNING

Qualifiers	Defaults
/BOUND	
/BREAK	
/CURRENT	/CURRENT
/END	/BEGINNING
/FORWARD	/CURRENT
/REVERSE	/CURRENT

Qualifiers

/BEGINNING (D)

Indicates that the cursor should be moved to the beginning of the line. The /BEGINNING, /BREAK, /BOUND, and /END qualifiers are mutually exclusive.

/BOUND

Moves the cursor to the beginning or the end of the current line, depending on whether the direction specified is FORWARD or REVERSE. If the cursor is already at the specified destination, LSE issues a message to that effect and the cursor does not move. The /BEGINNING, /BREAK, /BOUND, and /END qualifiers are mutually exclusive.

/BREAK

Moves the cursor either to the beginning or end of a line, depending on whether the direction currently specified is FORWARD or REVERSE. If the cursor is already at the specified destination, LSE moves it to the corresponding break on the next line in the current direction. The /BEGINNING, /BREAK, /BOUND, and/END qualifiers are mutually exclusive.

/CURRENT (D)

Instructs LSE to use the current direction of the buffer.

/END

Indicates that the cursor should be moved to the end of the line. The /BEGINNING, /BREAK, /BOUND, and /END qualifiers are mutually exclusive.

/FORWARD

Instructs LSE to move the cursor down, or to the right.

/REVERSE

Instructs LSE to move the cursor up, or to the left.

Description

The GOTO LINE command moves the cursor to one end of the line in the direction specified. If the cursor is already at the end of the current line, this command moves the cursor to the next line, unless you have specified the /BOUND qualifier.

Keypad Equivalent

GOTO LINE/BEGINNING/REVERSE

Key	Keypad Mode
Ctrl/H **BACKSPACE**	EDT LK201, EDT VT100
F12 **BOL**	EDT LK201

GOTO LINE/BEGINNING/CURRENT

Key	Keypad Mode
KP0 **LINE**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

GOTO LINE/END/CURRENT

Key	Keypad Mode
KP2 **EOL**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

GOTO LINE/BOUND/REVERSE

Key	Keypad Mode
Ctrl/H **BACKSPACE**	EVE LK201, EVE VT100
PF1- ←	EVE VT100, EVE LK201

GOTO LINE/BOUND/FORWARD

Key	Keypad Mode
Ctrl/E	EVE LK201, EVE VT100
PF1- >	EVE VT100, EVE LK201

GOTO LINE/BREAK/CURRENT

Key	Keypad Mode
F12 **MOVE BY LINE**	EVE LK201
Keypad minus (–) **MOVE BY LINE**	EVE VT100

Related Commands

GOTO CHARACTER

GOTO WORD

Example

```
LSE>
GOTO LINE/BOUND/REVERSE
```

Moves the cursor to the start of the current line. If the cursor is at the start of a line, LSE displays the message, “Already at the start of the line” when you enter this command.

GOTO MARK

GOTO MARK — Moves the cursor to a marker name defined by a SET MARK command.

Format

GOTO MARK marker-name

Parameter

marker-name

Specifies the name of a marker created with a SET MARK command.

Description

The GOTO MARK command moves the cursor to a marker name you define using a SETMARK command. LSE maps a new buffer to the current window if the marker you specify is not in the current buffer.

DECwindows Interface Equivalent

Pull-down menu: Navigate → Goto Mark

Related Commands

SET MARK

Example

```
LSE> GOTO MARK 1
```

Moves the cursor to the position previously marked using the command SET MARK 1.If MARK 1 is not in the current buffer, the buffer that contains MARK 1 becomes the current buffer.

GOTO PAGE

GOTO PAGE — Moves the cursor to the next page where a page boundary is a form feed, or the beginning or end of a buffer.

Format

GOTO PAGE

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT

Qualifiers	Defaults
/REVERSE	/CURRENT

Qualifiers

/CURRENT (D)

Instructs LSE to use the current direction of the buffer.

/FORWARD

Instructs LSE to move the cursor down.

/REVERSE

Instructs LSE to move the cursor up.

Description

The GOTO PAGE command moves the cursor to the beginning of the next or previous page in the current buffer, depending on the direction set by FORWARD or REVERSE. A form feed delimits a page. If there is no form feed in the current buffer, the GOTO PAGE command moves the cursor to the end (or beginning) of the buffer.

Keypad Equivalent

Key	Keypad Equivalent
KP7 **PAGE**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

Related Commands

`GOTO WORD`

GOTO PLACEHOLDER

GOTO PLACEHOLDER — Moves the cursor to a placeholder.

Format

`GOTO PLACEHOLDER`

Qualifiers	Defaults
/ALL	/ALL
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/NOPSEUDOCODE	
/REVERSE	/CURRENT

Qualifiers

/ALL (D)

Instructs the GOTO PLACEHOLDER command to recognize all placeholders, including pseudocode placeholders and overview records.

/CURRENT (D)

Instructs LSE to use the current direction of the buffer.

/FORWARD

Instructs LSE to move the cursor down, or to the right.

/NOPSEUDOCODE

Instructs the GOTO PLACEHOLDER command to ignore pseudocode placeholders.

/REVERSE

Instructs LSE to move the cursor up, or to the left.

Description

The GOTO PLACEHOLDER command moves the cursor to the next placeholder in the direction specified. A placeholder must be defined for the GOTO PLACEHOLDER command to recognize it.

Keypad Equivalent

GOTO PLACEHOLDER/ALL/FORWARD

Key	Keypad Mode
Ctrl/N	All

GOTO PLACEHOLDER/ALL/REVERSE

Key	Keypad Mode
Ctrl/P	All

GOTO PLACEHOLDER/NOPSEUDOCODE/FORWARD

Key	Keypad Mode
PF1-Ctrl/N	All

GOTO PLACEHOLDER/NOPSEUDOCODE/REVERSE

Key	Keypad Mode
PF1-Ctrl/P	All

Related Commands

`DEFINE PLACEHOLDER`

ERASE PLACEHOLDER

GOTO QUERY

GOTO QUERY — Moves the cursor to the specified SCA query session.

Format

GOTO QUERY query-name

Parameter

query-name

Specifies the name of the query session.

Description

The GOTO QUERY command splits the current window (if possible)and maps the specified query to the current window and the buffer associated with the query to the screen.

Related Commands

DELETE QUERY

FIND

NEXT QUERY

PREVIOUS QUERY

SHOW QUERY

Example

```
LSE> GOTO QUERY 1
```

Moves the cursor to the window containing query buffer 1.

GOTO REVIEW

GOTO REVIEW — Moves the cursor to the currently active review session.

Format

GOTO REVIEW

Description

The GOTO REVIEW command moves the cursor to the current review session and sets the current status to review mode. LSE maps the $REVIEW buffer to the screen and positions the cursor to the last current position in that buffer.

If no review session is currently active, the GOTO REVIEW command fails.

Related Commands

END REVIEW

GOTO QUERY

GOTO SOURCE

NEXT STEP

PREVIOUS STEP

REVIEW

GOTO SCREEN

GOTO SCREEN — Moves the cursor in the specified direction two lines less than the number of lines in the current window.

Format

GOTO SCREEN

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/REVERSE	/CURRENT

Qualifiers

/CURRENT (D)

Instructs LSE to use the current direction of the buffer.

/FORWARD

Instructs LSE to move the cursor down.

/REVERSE

Instructs LSE to move the cursor up.

Description

The GOTO SCREEN command moves the cursor two lines less than the number of lines in the current window, depending on the direction set by the /FORWARD or /REVERSE qualifier.

Users of the DECwindows interface can achieve similar results by pressingMB1 above or below the slider in the vertical scroll bar.

Keypad Equivalent

GOTO SCREEN/FORWARD

Key	Keypad Mode
E6 **NEXT SCREEN**	EDT LK201, EVE LK201
KP0 **Next Screen**	EVE VT100
None	EDT VT100

GOTO SCREEN/REVERSE

Key	Keypad Mode
E5 **PREV SCREEN**	EDT LK201, EVE LK201
Keypad period **Prev Screen**	EVE VT100
None	EDT VT100

GOTO SCREEN/CURRENT

Key	Keypad Mode
KP8 **SECT**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

GOTO SOURCE

GOTO SOURCE — Displays the source corresponding to the current diagnostic or query item. To display a query item, you must be using SCA.

Format

`GOTO SOURCE`

Qualifiers	
/READ_ONLY	
/WRITE	

Qualifiers

/READ_ONLY

Specifies that the buffer containing the source be set read-only and therefore unmodifiable. Using this qualifier overrides any setting established by the SET DIRECTORY command.

/WRITE

Specifies that the buffer containing the source be set writable and therefore modifiable. Using this qualifier overrides any setting established by the SET DIRECTORY command.

Description

The GOTO SOURCE command has different actions, depending on whether LSE is in review or query mode. To be in query mode, you must be using SCA.

Review Mode

In review mode, LSE selects the diagnostic at the current position in the buffer$REVIEW and a region where you want the source displayed. This becomes the current diagnostic.

LSE highlights the current diagnostic and the current region and displays in a second window, with the region highlighted, the file containing the current region. When a diagnostic is selected in this way, the buffer containing the current region becomes the current buffer.

LSE might display a suggested error correction and prompt for a yes (Y) or no (N) response; LSE makes the correction if you respond with a Y.

Query Mode

In query mode, LSE selects the query item occurrence at the current position in the current query buffer. This becomes the current query item. LSE highlights the current query item and displays the file containing the corresponding source for the current query item in a second window. The buffer containing the source that corresponds to the current query item becomes the current buffer.

Review or Query Modes

If the source file corresponding to the current diagnostic region or current query item is not in a buffer, LSE creates an unmodifiable buffer and reads the source file specified in the diagnostics file or SCA data file into that buffer.

If it cannot find that file, LSE uses the list of directories specified by the SET SOURCE_DIRECTORY command to find the file.

LSE uses CMS to access a file if the directory for the file to be accessed is the same as the translation of CMS$LIB.

Users of the DECwindows interface can invoke the GOTO SOURCE command by moving the mouse cursor to an occurrence in the query buffer, or an error region in the review buffer, and pressing MB1 twice.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl/G	All

DECwindows Interface Equivalent

Pop-up menu $\begin{Bmatrix}\text{Review buffer} \rightarrow \text{Goto Source} \\ \text{Query buffer} \rightarrow \text{Goto Source}\end{Bmatrix}$

Double click MB1 on the review or query item.

Pull-down menu: Source → Goto Source

Related Commands

SET DIRECTORY

SET SOURCE_DIRECTORY

SHOW DIRECTORY

SHOW SOURCE_DIRECTORY

CD

Example

```
LSE> GOTO SOURCE
```

Moves the cursor to the buffer containing the source code corresponding to the current diagnostic or query item.

GOTO TOP

GOTO TOP — Moves the cursor to the top of the current buffer.

Format

GOTO TOP

Description

The GOTO TOP command moves the cursor to the top of the buffer that contains the cursor. To achieve the same result, DECwindows interface users can use MB1 to drag the vertical scroll bar slider to the top of the scroll bar.

Keypad Equivalent

Key	Keypad Equivalent
PF1-KP5 **TOP**	EDT LK201, EDT VT100, EVE LK201
PF1-E5	EDT LK201
PF1-↑	EVE LK201, EVE VT100

DECwindows Interface Equivalent

Pull-down menu: Navigate → Goto Top

Related Commands

GOTO BOTTOM

GOTO WORD

GOTO WORD — Moves the cursor to the beginning of the current, next, or previous word in the current buffer, depending on the direction specified.

Format

GOTO WORD

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/REVERSE	/CURRENT

Qualifiers

/CURRENT (D)

Instructs LSE to use the current direction of the buffer.

/FORWARD

Instructs LSE to move the cursor down, or to the right.

/REVERSE

Instructs LSE to move the cursor up, or to the left.

Description

The GOTO WORD command moves the cursor to the first character of the current,next, or previous word, depending on the current direction or the direction set by the /FORWARD or /REVERSE qualifier. If the current direction is FORWARD, the cursor moves to the beginning of the next word. If the current direction is REVERSE, the cursor moves to the beginning of the current word; if the cursor is at the beginning of a word, it moves to the beginning of the previous word.

A word consists only of identifier characters and trailing blanks and can be delimited only by tabs or characters not specified in the /IDENTIFIER_CHARACTERS qualifier on the DEFINE LANGUAGE command. LSE also considers all nonblank, nonidentifier characters to be words.

Keypad Equivalent

Key	Keypad Equivalent
KP1 **WORD**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

Related Commands

GOTO CHARACTER

GOTO LINE

HELP

HELP — Displays information about LSE and SCA commands.

Format

HELP [topic-list]

Qualifiers	Defaults
/INDICATED	
/KEYPAD	
/LANGUAGE=language-name	
/LIBRARY=library-name	
/PACKAGE=package-name	

Qualifiers

/INDICATED

Causes LSE to display the help text associated with the token, placeholder, or routine at the current cursor position. If you do not specify or negate the /LANGUAGE qualifier or the /PACKAGE qualifier, LSE first looks for a language element. If the indicated item is not a language element, LSE looks for a package element.

The help text comes from the HELP library associated with the specified language or package. LSE forms a topic string by concatenating the /TOPIC_STRING qualifier associated with the language or package, followed by the indicated token, placeholder, or entry name. LSE then searches for the topic in the HELP library.

You cannot use the /INDICATED qualifier with any of the following qualifiers:/KEYPAD, /LANGUAGE, /LIBRARY, or /PACKAGE.

/KEYPAD

Specifies that you want keypad HELP. You cannot use the /KEYPAD qualifier with any of the following qualifiers: /INDICATED, /LANGUAGE, /LIBRARY, or /PACKAGE.

The /KEYPAD qualifier builds the keypad diagram by using legends specified with the /LEGEND qualifier on the DEFINE KEY command. When the diagram is displayed and you press a key, LSE looks up the topic specified for that key by using the/TOPIC qualifier on the DEFINE KEY command, and displays the corresponding help text. The HELP library accessed is LSE $KEYPAD.HLB.

/LANGUAGE=language-name

Causes LSE to take the value of the /TOPIC_STRING qualifier for the indicated language and concatenate that value to the front of the *topic-list* parameter on the HELP command. If you specify the /LANGUAGE qualifier without a value, LSE uses the language associated with the current buffer. (In this case, not having the current buffer associated with a language creates an error.)

You must not specify either the /KEYPAD qualifier or the /PACKAGE qualifier with the /LANGUAGE qualifier.

/LIBRARY=library-name

Specifies which HELP library LSE searches for the topic. This qualifier overrides the library file determined by the default behavior of LSE. You can specify any other qualifiers with the /LIBRARY qualifier except for the /KEYPAD qualifier.

/PACKAGE=package-name

Causes LSE to take the value of the /TOPIC_STRING qualifier for the indicated package and concatenate that value to the front of the *topic-list* parameter on the HELP command. You must provide the package name as the value of the qualifier.

You must not specify either the /KEYPAD or the /LANGUAGE qualifier with the/PACKAGE qualifier.

Parameter

topic-list

Indicates the topic for which you want help. This can be any list of topics valid for input to the DCL command interpreter's HELP command. The topic list must not be specified with the /INDICATED qualifier.

Description

The HELP command displays information about the requested topic of LSE, a language, or a package.

If you have more than one screen of help text available, and do not want to review the additional screens of information, press Ctrl/Z to return to editing mode.

After exiting from HELP, the buffer $HELP contains the text displayed by the HELP command. This does not happen if you are using keypad HELP.

Keypad Equivalent

HELP/KEYPAD (VT100 keypad)

Key	Keypad Mode
PF2 **HELP**	All

HELP/KEYPAD (VT200 keypad or higher)

Key	Keypad Mode
Help	EDT LK201, EVE LK201

HELP/INDICATED

Key	Keypad Mode
PF1-PF2 **HELP IND**	All

Key	Keypad Mode
PF1-Help	EDT LK201, EVE LK201

Examples

1. `LSE>` ***HELP CREATE LIBRARY***

   Invokes HELP at the LSE level.

2. `LSE>` ***HELP/LANGUAGE=PASCAL STATEMENTS***

   Indicates that, for Pascal, the value PASCAL is assigned to the /TOPIC_STRING qualifier. LSE HELP is invoked to provide information about the STATEMENTS topic list.

IMPORT

IMPORT — Performs a conversion of XREF files into analysis data files.

Format

IMPORT file-spec[, . . .]

Qualifiers	Defaults
/[NO]LOG	/LOG
/OUTPUT[=file-spec]	

Qualifiers

/LOG (D)
/NOLOG

Indicates whether SCA reports successful file conversions.

/OUTPUT[=file-spec]

Specifies that file conversion data is to go to a file rather than be displayed on your screen (or go to a batch log file). The default output file specification is SCA.LIS.

Parameter

file-spec

Specifies the XREF files to be converted to SCA analysis data files. Wildcards can be used, and the default extension is .XREF.

Description

The VSI C++ compilers do not generate analysis data files that can be directly loaded into an SCA library. Instead, they generate XREF data files, which must be converted using the SCA IMPORT command. Future versions of other compilers might also require the use of this command.

Example

```
LSE> IMPORT PHASE.XREF NEWPHASE.ANA
```

Converts an XREF file specification to an ANA analysis data file with a different file name.

INCLUDE

INCLUDE — Inserts the specified file at the current editing position.

Format

INCLUDE file-spec

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Specifies a buffer into which the file is to be included. If the buffer does not exist, it is created for display only (the buffer cannot be written back to a file).

Parameter

file-spec

Specifies the file to be copied to the current editing position. Wildcards are permitted in DECwindows mode.

Description

The INCLUDE command inserts the contents of the specified file at the current editing position. After inserting the file, the editing cursor is positioned on the first character of the inserted text.

This command is similar to the READ command, except that the INCLUDE command inserts the file's contents into the receiving buffer at the position your cursor was on. The cursor is then positioned on the first character of the inserted text, rather than remaining on the original character.

DECwindows Interface Equivalent

Pop-up menu: None

Pull-down menu: File → Include File . . .

Example

```
LSE> INCLUDE y.x
```

Opens file *y.x* for input and inserts its contents at the current editing position, which leaves the cursor on the first character of the inserted text.

INSPECT

INSPECT — Inspects the consistency between declarations or references for the same symbol.

Format

`INSPECT query-expression`

Qualifiers	Defaults
/CHARACTERISTICS=(option[. . .])	/CHARACTERISTICS=ALL
/DESCRIPTION=string	
/[NO]DISPLAY[=(option, . . .)]	/DISPLAY=DEFAULT
/[NO]ERROR_LIMIT=(global-limit[,symbol-limit])	/NOERROR_LIMIT
/[NO]LOG	/LOG
/[NO]MODIFY[=query-name]	/NOMODIFY
/NAME=query-name	
/OUTPUT[=file-spec]	
/[NO]REPLACE	/NOREPLACE
/[NO]RESULT=option	/RESULT=DEFAULT
/SEVERITY_LEVEL=severity-level	/SEVERITY=INFORMATIONAL
/[NO]SYNCHRONIZE	/NOSYNCHRONIZE

Qualifiers

/CHARACTERISTICS=(option[…])
/CHARACTERISTICS=ALL (D)

Indicates which characteristics of the occurrences should be checked. You can use one or more of the following options to request specific information:

Option	Description
IMPLICIT_DECLARATIONS	Checks that all symbols are explicitly declared
TYPE	Checks that the types of all occurrences of each symbol match
UNIQUENESS	Checks that multiple declarations of the same symbol have the same name
UNUSED_SYMBOLS	Checks that all symbols are used
USAGE	Looks for symbols that are read but never written, or written but never read
ALL	Checks all of the preceding characteristics

Any of these options (except ALL) can have the prefix NO to indicate that the characteristic should not be checked.

Each of the characteristic options takes a *query-expression* as an optional value. The characteristic-specific query expression specifies the set of occurrences for which that characteristic will be checked. If the prefix NO is present, the query expression indicates occurrences for which that characteristic will not be checked. The default query expression for each characteristic option is to check all occurrences.

/DESCRIPTION=string

Specifies a single line of text that is displayed along with the query name when the query is displayed by entering the SHOW QUERY command.

/DISPLAY[=(option, . . .)]
/DISPLAY=DEFAULT (D)
/NODISPLAY

Indicates how much information SCA displays about query results. Use one or more of the following keywords to request specific information:

Keyword	Description
NAME	Symbol name
CLASS	Class of item
LINE_NUMBER	Compilation line number
FILE_NAME	File name and type containing a symbol occurrence
FULL_FILE_SPEC	Complete file specification containing a symbol occurrence
RECORD_NUMBER	Record number within a source file
OCCURRENCE_TYPE	Type of symbol occurrence (such as declaration, read, or call)
ALL	All of the previous options
DEFAULT	Default settings of the display options
NONE	Nothing (equivalent to the /NODISPLAY qualifier)

You can prefix any keyword (except ALL, DEFAULT, and NONE) with NO to request that information be excluded.

The initial default for each type of new query is as follows:

```
DISPLAY=(NAME,CLASS,MODULE,LINE_NUMBER,OCCURRENCE_TYPE)
```

/ERROR_LIMIT=(global-limit[,symbol-limit])
/NOERROR_LIMIT (D)

Specifies the maximum number of errors that the INSPECT command should report. This causes the INSPECT command to stop if the number of errors exceeds the maximum.

The *global-limit* parameter specifies the maximum number of errors reported for all symbols before the INSPECT command stops.

The *symbol-limit* parameter specifies the maximum number of errors reported for a particular symbol before the INSPECT command stops reporting errors for that symbol.

/LOG (D)
/NOLOG

Indicates whether the count of symbol occurrences will be reported.

/MODIFY[=query-name]
/NOMODIFY (D)

Indicates that an existing query is to be modified. By default,each INSPECT command creates a new query.

The /MODIFY=query-name qualifier indicates that the specified query should be modified according to the specification of the INSPECT command. The specified query must already exist.

By default, the /MODIFY qualifier specifies the current query.

/NAME[=query-name]

Specifies the name of the query. If a query with the same name already exists, you must also specify the /REPLACE qualifier. If a query name is not specified, SCA assigns a unique name to the query.

/OUTPUT[=file-spec]

Specifies that command output is to go to a file rather than be displayed on your screen (or go to a batch log file). The default output-file specification is SCA.LIS.

/REPLACE
/NOREPLACE (D)

Indicates whether existing queries should be replaced by new queries. By default, an INSPECT command that creates a query with the same name as an already existing query will fail.

/RESULT=option
/RESULT=DEFAULT (D)
/NORESULT

Indicates the type of query results displayed. You must specify one of the following keywords:

Keyword	Description
SYMBOLS	Only symbols are displayed.
OCCURRENCES	Symbols and occurrences are displayed.
DEFAULT	Either symbols or occurrences, or both, aredisplayed. SCA chooses the result type that is most appropriatefor the current query.

The /NORESULT qualifier specifies that no results should be displayed. This means that no query evaluation is done. If a query result exists because you entered an INSPECT command,specifying /NORESULT causes that result to be deleted.

/SEVERITY_LEVEL=severity-level
/SEVERITY=INFORMATIONAL (D)

Indicates the lowest severity level for diagnostics to be reported, as follows:

INFORMATIONAL
WARNING
ERROR
FATAL_ERROR

/SYNCHRONIZE
/NOSYNCHRONIZE (D)

Indicates that the query result must be synchronized with the current state of the virtual library being queried. By default,/NOSYNCHRONIZE causes SCA to do as little processing as necessary to evaluate the query. This can lead to query results that reflect the state of the virtual library at the time of a previous query.

The /SYNCHRONIZE qualifier specifies that the query result must be synchronized with the current virtual library. SCA attempts to minimize the amount of processing, but the result is still synchronized with the virtual library that was in effect at the time the query was evaluated.

Parameter

query-expression

Specifies the set of occurrences to be inspected.

Description

The INSPECT command checks the consistency between declarations or references for the same symbol.

Related Commands

FIND

Example

```
LSE> INSPECT *
```

Inspects all characteristics of all symbols.

LINE

LINE — Moves the cursor in the current buffer to the start of the source line you specify.

Format

LINE integer [procedure-name]

Parameters

integer

Specifies the number of the line in the current buffer to which you want LSE to move the cursor. If you do not specify a line number, LSE prompts for one. Pressing Ctrl/Z at the prompt cancels the command.

procedure-name

Specifies the name of a DECTPU procedure in the current buffer. The procedure name is valid only for DECTPU source files. This parameter is useful because some compiler messages refer to line numbers in a procedure.

To find out the current line number and total number of lines in the buffer, use the WHAT LINE command.

Description

The LINE command moves the cursor in the current buffer to the start of the line you specify. If the line requested is hidden, the overview records are expanded to the source level and the cursor is placed on there quested line.

Related Commands

WHAT LINE

Examples

1. LSE> ***LINE 14***

   Moves the cursor to the beginning of line 14.

2. LSE> ***LINE 12 user_proc***

   Moves the cursor to the beginning of line 12 of a procedure named *user_proc*.

LOAD

LOAD — Loads one or more files of compiler-generated, source-analysis data into an SCA library.

Format

LOAD file-spec[, . . .]

Qualifiers	Defaults
/[NO]DELETE	NODELETE
/LIBRARY=library-spec	/LIBRARY=primary-library
/[NO]LOG	/LOG
/[NO]REPLACE	/REPLACE

Qualifiers

/DELETE
/NODELETE (D)

Deletes an analysis data file after it has been successfully loaded into an SCA library.

/LIBRARY=library-spec
/LIBRARY=primary-library (D)

Specifies an SCA physical library to update. This library must be one of the current SCA libraries established by a SET LIBRARY command.

If you do not specify this qualifier, SCA refers to the primary SCA library; that is, SCA updates the first of the current SCA physical libraries.

/LOG (D)
/NOLOG

Indicates whether SCA reports successful updating of SCA libraries.

/REPLACE (D)
/NOREPLACE

Indicates whether SCA replaces existing modules of source analysis data with new information.

Parameter

file-spec[, . . .]

Specifies one or more files of source-analysis data to be loaded into an SCA library. You can use a wildcard file specification.

The default file type is .ANA, which is the default file type for source-analysis data files created by compilers.

Description

With the LOAD command, you can load SCA library files with compiler-generated source information.

Related Commands

SET LIBRARY

Example

```
$ SCA LOAD obj:getfile*
```

Loads the specified modules, located at a directory defined as obj,into the current library.

For additional examples, see the section about loading a library in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

LOWERCASE WORD

LOWERCASE WORD — Changes the letters in the current word or the selected range to lowercase.

Format

LOWERCASE WORD

Description

The LOWERCASE WORD command changes the letters in the current word to lowercase. If the word contains both uppercase and lowercase characters, LSE changes all letters to lowercase.

If the cursor is between words, LSE changes the following word to lowercase. If a selected range is active, all the words within that range are changed to lowercase. The cursor then moves to the start of the next word.

DECwindows Interface Equivalent

Pull-down menu: Edit → Lowercase

Related Commands

CAPITALIZE WORD

UPPERCASE WORD

MODIFY LANGUAGE

MODIFY LANGUAGE — Modifies the characteristics of the specified language.

Format

MODIFY LANGUAGE language-name

Qualifiers	Default
/BOOK=file-spec, defined_language	
/CAPABILITIES=[NO]DIAGNOSTICS	
/COMMENT=(specifier, . . .)	
/COMPILE_COMMAND=string	
/EXPAND_CASE=AS_IS	
/EXPAND_CASE=LOWER	
/EXPAND_CASE=UPPER	
/FILE_TYPES=(file-type[, . . .])	
/FORTRAN=[NO]ANSI_FORMAT	
/[NO]HELP_LIBRARY=file-spec	
/IDENTIFIER_CHARACTERS=string	
/INITIAL_STRING=string	
/LEFT_MARGIN= *n*	/LEFT_MARGIN=1
/LEFT_MARGIN=CONTEXT_DEPENDENT	
/OVERVIEW_OPTIONS=(MINIMUM_LINES= *m*, TAB_RANGE= (t1,t2))	

Qualifiers	Default
/PLACEHOLDER_DELIMITERS= (delimiter-specification[, …])	
/PUNCTUATION_CHARACTERS=string	
/[NO]QUOTED_ITEM= (QUOTES=string[,ESCAPES=string])	
/REFERENCE=file-spec, defined_language	
/RIGHT_MARGIN= *n*	
/TAB_INCREMENT= *n*	
/TOPIC_STRING=string	
/VERSION=string	
/[NO]WRAP	

Qualifiers

/BOOK=file-spec, defined_language

Specifies the default online-book file name, defining the book LSE uses to retrieve online text for a placeholder or token whose book is undefined.

/CAPABILITIES=DIAGNOSTICS
/CAPABILITIES=NODIAGNOSTICS

Specifies whether the compiler can generate diagnostic files.

/COMMENT=(specifier, . . .)

Specifies the character sequences of comments in the language. The specifiers are as follows:

- ASSOCIATED_IDENTIFIER=keyword

  Indicates the preferred association of comments to identifier. You can specify one of the following values:

  - NEXT – Indicates that comments should be associated with the next identifier
  - PREVIOUS – Indicates that comments should be associated with the preceding identifier

- BEGIN=list of quoted strings

  END=list of quoted strings

  Defines the character sequences that start and end bracketed comments. A bracketed comment begins and ends with explicit comment delimiters. (Note that the beginning and ending comment delimiters can be the same, but need not be.) The list provided with the specifiers BEGIN and END can be any of the following:

  - A string that is the one open comment sequence for the language. You must enclose this in quotes.
  - A parenthesized list of strings, each one of which can be an open comment sequence for the language. You must enclose each one in quotes.

The list accompanying the BEGIN specifier must be consistent with the list accompanying the END specifier. If the BEGIN specifier lists a string, the END specifier must also list a string.

Bracketed comments are recognized by the formatting commands (see the ALIGN and FILL commands) and placeholder operations (see the ERASEPLACEHOLDER command and the /DUPLICATION qualifier of the DEFINEPLACEHOLDER command).

- TRAILING=list of quoted strings Defines the character sequence that introduces line-oriented comments. Aline-oriented comment begins with a special character sequence (consisting of one or more characters) and ends at the end of the line. The list provided with the TRAILING specifier can be any of the following:

  - A string that is the one-line comment sequence for the language
  - A list of strings enclosed in parentheses; each string can be a line-comment sequence for the language

  Line comments are recognized by the formatting commands and placeholder operations, just as bracketed comments are.

- LINE=list of quoted strings

  Requires that the comment delimiter be the first character that is not blank on the line. The LINE specifier is particularly useful with block comments, such as the following:

```
/*
** Here is the inside of a comment
** which has LINE="**" specified
*/
```

- FIXED=quoted string, column number

  Used for languages that require that a specific comment delimiter be placed in a specific column, such as FIXED=("* ",1) for COBOL.

/COMPILE_COMMAND=string

Specifies the default command string for the COMPILE command. (See the explanation of the *command-string* parameter in the COMPILE command entry.)

/EXPAND_CASE=AS_IS
/EXPAND_CASE=LOWER
/EXPAND_CASE=UPPER

Specifies the case of the text of the inserted template. The value AS_IS specifies that the inserted template be expanded according to the case in the token or placeholder definition. The values LOWER and UPPER specify that the inserted template be expanded in lowercase or uppercase, respectively.

/FILE_TYPES=(file-type[, . . .])

Specifies a list of file types that are valid for the language being defined. The file types must be enclosed in quoted strings. When LSE reads a file into a buffer, it sets the language for that buffer automatically if it recognizes the file type. For example, a Fortran file type (.FOR) sets the language to Fortran. The period character must be included with the file type.

/FORTRAN=ANSI_FORMAT
/FORTRAN=NOANSI_FORMAT

Specifies special processing for ANSI Fortran. Note that some commands behave differently when you use the /FORTRAN qualifier. Specifying NOANSI_FORMAT causes LSE to insert templates in non-ANSI (tab) format.

/HELP_LIBRARY=file-spec
/NOHELP_LIBRARY

Specifies the HELP library where you can find help text for placeholders and tokens defined in this language. LSE applies the default file specification SYS$HELP:HELPLIB.HLB. If you want to access some HELP library other than SYS$HELP, you must supply an explicit device name.

/IDENTIFIER_CHARACTERS=string

Specifies the characters that can appear in token and alias names in that language. This list of characters is used in various contexts for the/INDICATED qualifier.

The list of identifier characters also determines what LSE considers to be a word. A word is a sequence of identifier characters, possibly followed by one or more blanks. All nonblank, nonidentifier characters are considered to be distinct words.

If you do not specify the /IDENTIFIER_CHARACTERS qualifier, LSE supplies the following values by default:

```
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ%$_0123456789"
```

/INITIAL_STRING=string

Specifies the initial text to appear in a newly created buffer.

/LEFT_MARGIN=n
/LEFT_MARGIN=1 (D)
/LEFT_MARGIN=CONTEXT_DEPENDENT

Specifies the left margin setting to be associated with the language.

If you specify CONTEXT_DEPENDENT as the column number, LSE uses the indentation of the current line to determine the left margin when you use the /WRAP qualifier. When you use the FILL command, LSE uses the indentation of the first line of each selected paragraph to determine the left margin.

/OVERVIEW_OPTIONS=(MINIMUM_LINES=m, TAB_RANGE= (t1,t2))

Specifies both the minimum number of lines an overview line must hide and the range of acceptable tab increments.

The specifiers are as follows:

- MINIMUM_LINES= *m*

  Specifies the minimum number of lines an overview line must hide. The default is 1. For example, if the value of the parameter on MINIMUM_LINES is 5, a line hides other lines only if there are at least 5 lines to hide. This specifier helps the user to avoid having very small source-line groups,which avoids many expansion levels.

- TAB_RANGE=(t1,t2)

  The TAB_RANGE specifier indicates the range of tab values for which the adjustment definitions are valid. The default is (4,8). The second value must be at least twice the first value; both values must be positive. For example, if the tab range is (4,8), LSE assumes that the adjustment definitions will work for any DEFINE LANGUAGE/TAB_INCREMENT value from 4 to 8, inclusive. If you specify a /TAB_INCREMENT value outside the tab range, LSE recomputes indentation to make the adjustments work.

  For best performance, it is recommended that you avoid recomputation by choosing a range that covers reasonable values. The numbers specified for the DEFINE ADJUSTMENT/CURRENT and DEFINE ADJUSTMENT/SUBSEQUENT commands must work for any tab increment value in the tab range.

/PLACEHOLDER_DELIMITERS=(delimiter-specification[, . . .])

Specifies the starting and ending strings that delimit placeholders. Placeholders can specify single constructs or lists of constructs. The delimiters for each type of placeholder are specified as a pair of quoted strings separated by commas and enclosed in parentheses.

The format of a delimiter specification is as follows:

```
keyword=(starting-string,ending-string)
```

Possible keywords are REQUIRED, REQUIRED_LIST, OPTIONAL, OPTIONAL_LIST, or PSEUDOCODE. If you do not use the PSEUDOCODE keyword, the default is NOPSEUDOCODE. The maximum length of these strings is seven characters.

The following is an example of a complete set of placeholder delimiter specifications:

```
/PLACEHOLDER_DELIMITERS = ( -
 REQUIRED =("{<",">}"), -
 REQUIRED_LIST=("{<",">}..."), -
 OPTIONAL =("[<",">]"), -
 OPTIONAL_LIST=("[<",">]..."), -
 PSEUDOCODE=("«" , "»"))
```

If any of the five keywords are not specified with the /PLACEHOLDER_DELIMITERS qualifier, LSE applies the following defaults:

```
/PLACEHOLDER_DELIMITERS = ( -
 REQUIRED =("{","}"), -
 REQUIRED_LIST=("{","}..."), -
 OPTIONAL =("[","]"), -
 OPTIONAL_LIST=("[","]..."), -
 NOPSEUDOCODE)
```

/PUNCTUATION_CHARACTERS=string

Specifies the characters considered punctuation marks, or delimiters,in the language. When a placeholder name and its enclosing brackets are deleted, preceding white space is also deleted if there are punctuation characters to delimit the program constructs.

/QUOTED_ITEM=(QUOTES=string [,ESCAPES=string])
/NOQUOTED_ITEM

Describes the syntax of certain language elements, such as strings, that require special handling for proper text formatting. LSE uses the /QUOTED_ITEM qualifier to detect comments properly. LSE

does not acknowledge comment strings that occur within quoted items, nor does LSE acknowledge quoted elements that occur within comments.

The value of the /QUOTED_ITEM qualifier indicates the syntax of a quoted item. This value must be a keyword list. The keywords are as follows:

- QUOTES

  This keyword is required and must have an explicit value. The value must be a quoted string denoting all the quote characters in the language. LSE assumes that quoted items begin and end with the same character.

- ESCAPES

  This keyword is optional. If given, the value is required and must be a quoted string containing the escape characters for quoted items. Some languages use escape characters to insert quoting characters into strings. For example, C uses the backslash (\) as an escape character. If you omit this keyword, LSE assumes that the language inserts quote characters into strings by doubling them.

/REFERENCE=book_reference, defined_language

Specifies the book-reference tag string, defining the section of a book to display for a placeholder or token whose reference tag is undefined.

/RIGHT_MARGIN=n

Specifies the right margin setting to be associated with the language. By default, the right margin is set at column 80.

/TAB_INCREMENT=n

Specifies that tab stops be set every *n* columns beginningwith column 1.

/TOPIC_STRING=string

Specifies a prefix string to be concatenated to the /TOPIC_STRING qualifier specified in a placeholder or token definition before LSE looks up the help text for that placeholder or token. (Typically, this is the name of the language in the HELP library.)

/VERSION=string

Specifies a string that represents the version number of the tokens and placeholders associated with this language. Use the SHOWLANGUAGE command to display this string.

/WRAP
/NOWRAP

Specifies whether the ENTER SPACE command (bound to the space bar by default) should wrap text when there is too much to fit on the current line. The /NOWRAP qualifier disables text wrapping.

Parameter

language-name

Specifies the name of the language whose characteristics are to be defined.

Description

With the MODIFY LANGUAGE command, you can supersede text characteristics that you have set for a specific language. It does not affect other characteristics that you might have changed from the initial default by using the DEFINE LANGUAGE command.

Related Commands

DEFINE LANGUAGE

DELETE LANGUAGE

EXTRACT LANGUAGE

SET LANGUAGE

SHOW LANGUAGE

Examples

```
1.  LSE> MODIFY LANGUAGE SAMPLE /EXPAND_CASE=LOWER
```

Makes every letter lowercase in the template for the language SAMPLE; this includes the words inside comments.

```
2.  LSE> MODIFY LANGUAGE FORTRAN /FORTRAN=ANSI_FORMAT
```

Sets ANSI_FORMAT as the format for your Fortran language definition.

```
3.  LSE> MODIFY LANGUAGE Ada /PLACEHOLDER_DELIMITERS=PSEUDOCODE=("«" , "»")
```

Sets pseudocode placeholder delimiters for Ada.

NEXT BUFFER

NEXT BUFFER — Moves your next buffer into the current window, which returns you to your last position in that buffer.

Format

NEXT BUFFER

Description

The NEXT BUFFER command moves the cursor to the next buffer in the list of buffers and maps that buffer to the current window. This allows you to cycle through several buffers without having to type their names.

If you have only two buffers, repeating NEXT BUFFER toggles between them. If you have more than two buffers, the next buffer is determined by the order in which you created the buffers. Only user buffers are included in the list of buffers. For a list of your buffers, enter the SHOW BUFFER/USER_BUFFERS command.

If you enter a NEXT BUFFER command while you are positioned in the last buffer in the list, LSE takes you to the first buffer in the list.

Users of the DECwindows interface can press MB1 with the mouse cursor on the buffer name to cycle through the user buffers.

DECwindows Interface Equivalent

User buffer status line: → Buffer name

Related Commands

GOTO BUFFER

PREVIOUS BUFFER

SHOW BUFFER

Example

```
LSE> NEXT BUFFER
```

Moves your next buffer into the current window.

NEXT ERROR

NEXT ERROR — Selects the next diagnostic in the current set of diagnostics.

Format

NEXT ERROR

Description

The NEXT ERROR command positions the cursor at the next diagnostic in the buffer$REVIEW, which contains the current set of diagnostics. If the current error is the last in the set, the NEXT ERROR command does not wrap around from the last error back to the first.

If you are in review mode, a NEXT STEP command is equivalent to a NEXT ERROR command.

DECwindows Interface Equivalent

Pop-up menu: Review buffer → Next Error

Pull-down menu: Source → Next Error

Related Commands

GOTO REVIEW

NEXT STEP

PREVIOUS ERROR

REVIEW

NEXT OCCURRENCE

NEXT OCCURRENCE — Moves the cursor forward to the next occurrence of the current source symbol in the current query and highlights that next occurrence.

Format

NEXT OCCURRENCE

Description

The NEXT OCCURRENCE command moves the cursor forward to the next occurrence in the current query; that occurrence is highlighted. If there are no more occurrences of the current source symbol, LSE interprets the command as a NEXT SYMBOL command. If necessary, LSE remaps the query buffer.

DECwindows Interface Equivalent

Pop-up menu: Query buffer > Next Occurrence

Related Commands

GOTO REVIEW

NEXT STEP

PREVIOUS ERROR

REVIEW

NEXT QUERY

NEXT QUERY — Moves the cursor to the next SCA query session.

Format

NEXT QUERY

Description

The NEXT QUERY command moves the cursor to the next session in a series of SCA query sessions. LSE maps the query display and moves the cursor to the last remembered position in that query. SCA determines the order of multiple query sessions by the order in which the sessions were created.

DECwindows Interface Equivalent

Query buffer status line: → Query-name

Related Commands

NEXT STEP

NEXT SYMBOL

PREVIOUS OCCURRENCE

NEXT STEP

NEXT STEP — Moves the cursor forward to the next error, item, or occurrence, depending on whether LSE is in review or query mode. The specified item is highlighted.

Format

NEXT STEP

Description

The NEXT STEP command moves the cursor in a manner that depends on the current mode:

- In review mode, LSE treats this command as a NEXT ERROR command.
- In query mode, the NEXT STEP command moves the cursor to the next line in the query display and highlights it, whether it is a symbol or an occurrence.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl/F	All

Related Commands

NEXT ERROR

NEXT NAME

NEXT OCCURRENCE

NEXT SYMBOL

PREVIOUS STEP

NEXT SYMBOL

NEXT SYMBOL — Moves the cursor forward to the next source symbol in the current query and highlights that next symbol.

Format

NEXT SYMBOL

Description

The NEXT SYMBOL command moves the cursor forward to the next source symbol in the current query and highlights this symbol. If necessary, LSE remaps the query buffer.

DECwindows Interface Equivalent

Pop-up menu: Query buffer → Next Symbol

Related Commands

NEXT STEP

PREVIOUS SYMBOL

NEXT WINDOW

NEXT WINDOW — Moves the cursor from the current window to the next window, if the screen is split into multiple windows.

Format

NEXT WINDOW

Description

The NEXT WINDOW command works only if the screen displays multiple windows. LSE positions the cursor in the next window on the screen.

NEXT WINDOW is synonymous with the OTHER WINDOW command.

Keypad Equivalent

Key	Keypad Equivalent
PF1- ↓ **NXT WNDW**	EDT LK201, EDT VT100
PF1-E6 **NXT WNDW**	EVE LK201

Related Commands

CHANGE WINDOW_MODE

DELETE WINDOW

ENLARGE WINDOW

ONE WINDOW

OTHER WINDOW

PREVIOUS WINDOW

SET SCREEN

SHRINK WINDOW

TWO WINDOWS

ONE WINDOW

ONE WINDOW — Deletes all windows but the current window.

Format

ONE WINDOW

Description

The ONE WINDOW command removes from your screen all windows associated with your current editing session, except the one that currently has input focus.

DECwindows Interface Equivalent

Pull-down menu: View → One Window

Related Commands

CHANGE WINDOW_MODE

DELETE WINDOW

ENLARGE WINDOW

OTHER WINDOW

PREVIOUS WINDOW

SET SCREEN

SHRINK WINDOW

TWO WINDOWS

OTHER WINDOW

OTHER WINDOW — Moves the cursor from the current window to the next window, if the screen is split into multiple windows.

Format

OTHER WINDOW

Description

The OTHER WINDOW command works only if the screen displays multiple windows. LSE positions the cursor in the next window on the screen.

The OTHER WINDOW command is synonymous with the NEXT WINDOW command.

Related Commands

CHANGE WINDOW_MODE

DELETE WINDOW

ENLARGE WINDOW

ONE WINDOW

PREVIOUS WINDOW

SET SCREEN

SHRINK WINDOW

TWO WINDOWS

PASTE

PASTE — Copies the contents of the specified buffer into the current buffer at the current cursor position.

Format

PASTE

Qualifiers	Defaults
/BUFFER=buffer-name	/BUFFER=$PASTE (D)
/CLIPBOARD	See text

Qualifiers

/BUFFER=buffer-name
/BUFFER=$PASTE (D)

Specifies the buffer to be copied into the current buffer.

/CLIPBOARD

Instructs LSE to use the DECwindows clipboard, instead of a buffer, to supply the text being inserted. The /CLIPBOARD and /BUFFER qualifiers are mutually exclusive.

Description

The PASTE command copies text from a specified location to the current buffer. If you do not specify a buffer to copy from, LSE copies from the location(DECwindows Clipboard or character-cell terminal $PASTE buffer) that contains the text you last removed using the CUT command.

For users of the DECwindows interface, the default setting is /CLIPBOARD;otherwise, the default is /BUFFER=$PASTE.

Keypad Equivalent

Key	Keypad Equivalent
PF1-KP6 **PASTE**	EDT LK201, EDT VT100, EVE LK201
E2 **INSERT HERE**	EDT LK201, EVE LK201
KP9 **INSERT HERE**	EVE VT100

DECwindows Interface Equivalent

Pop-up menu: User buffer → Paste

Pull-down menu: Edit → Paste

Related Command

CUT

PREVIOUS BUFFER

PREVIOUS BUFFER — Moves your previous buffer into the current window, which returns you to your last position in that buffer.

Format

PREVIOUS BUFFER

Description

The PREVIOUS BUFFER command moves the cursor back to the previous buffer in the list of buffers and maps that buffer to the current window. This allows you to cycle through several buffers without having to type their names.

If you have only two buffers, repeating PREVIOUS BUFFER toggles between them. If you have more than two buffers, the previous buffer is determined by the order in which you created the buffers. Only user buffers are included in the list of buffers. For a list of your buffers, use the SHOW BUFFER/USER_BUFFERS command.

If you enter a PREVIOUS BUFFER command while you are positioned in the first buffer in the list, LSE takes you to the last buffer in the list.

Related Commands

GOTO BUFFER

NEXT BUFFER

SHOW BUFFER

Example

```
LSE> PREVIOUS BUFFER
```

Moves your previous buffer into the current window.

PREVIOUS ERROR

PREVIOUS ERROR — Selects the previous diagnostic in the current set of diagnostics.

Format

PREVIOUS ERROR

Description

The PREVIOUS ERROR command positions the cursor at the previous diagnostic in the buffer $REVIEW, which contains the current set of diagnostics. If the current error is the first in the set, the PREVIOUS ERROR command does not wraparound from the first error backwards to the last. If necessary, LSE remaps the $REVIEW buffer.

If you are in review mode, a PREVIOUS STEP command is equivalent to a PREVIOUSERROR command.

DECwindows Interface Equivalent

Pop-up menu: Review buffer → Previous Error

Pull-down menu: Source → Previous Error

Related Commands

GOTO REVIEW

NEXT ERROR

PREVIOUS STEP

REVIEW

PREVIOUS OCCURRENCE

PREVIOUS OCCURRENCE — Moves the cursor back to the previous occurrence of the current source symbol in the current query and highlights that occurrence.

Format

PREVIOUS OCCURRENCE

Description

The PREVIOUS OCCURRENCE command moves the cursor back to the previous occurrence in the current query; that occurrence is highlighted. If there are no more occurrences of current source symbols, LSE interprets the command as a PREVIOUS ITEM command. If necessary, LSE remaps the query.

DECwindows Inter

Pop-up menu: Query buffer → Previous Occurrence

Related Commands

NEXT OCCURRENCE

PREVIOUS ITEM

PREVIOUS STEP

PREVIOUS QUERY

PREVIOUS QUERY — Moves the cursor back to the previous SCA query session.

Format

PREVIOUS QUERY

Description

The PREVIOUS QUERY command moves the cursor back to the previous session in a series of SCA query sessions. LSE maps the query display and moves the cursor to the last remembered position in that query. SCA determines the order of multiple query sessions by the order in which the sessions were created.

Related Command

DELETE QUERY

GOTO QUERY

NEXT QUERY

PREVIOUS STEP

PREVIOUS STEP — Moves the cursor back to the previous error, item, name, or occurrence,depending on whether LSE is in review or query mode. That item is highlighted.

Format

PREVIOUS STEP

Description

The PREVIOUS STEP command moves the cursor in a manner that depends on the current mode:

- In review mode, LSE treats this command as a PREVIOUS ERROR command.

- In query mode, the PREVIOUS STEP command moves the cursor to the previous line and highlights it, whether it is a symbol or occurrence.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl/B	All

Related Commands

PREVIOUS ERROR

PREVIOUS ITEM

PREVIOUS OCCURRENCE

NEXT STEP

PREVIOUS SYMBOL

PREVIOUS SYMBOL — Moves the cursor back to the previous source symbol in the current query and highlights that source symbol.

Format

PREVIOUS SYMBOL

Description

The PREVIOUS SYMBOL command moves the cursor back to the previous source symbol in the current query; that source symbol is highlighted. If no more source symbols with the current name exist, LSE interprets the command as a PREVIOUSNAME command. If necessary, LSE remaps the query.

DECwindows Interface Equivalent

Pop-up menu: Query buffer → Previous Symbol

Related Commands

NEXT SYMBOL

PREVIOUS STEP

PREVIOUS WINDOW

PREVIOUS WINDOW — Moves the cursor from one window to the previous window, if the screen is split into multiple windows.

Format

PREVIOUS WINDOW

Description

The PREVIOUS WINDOW command moves the cursor from the bottom window to the top, in sequence, if the screen is split into multiple windows.

Keypad Equivalent

Key	Keypad Equivalent
PF1-↑ **PRV WNDW**	EDT LK201, EDT VT100
PF1-E5 **PRV WNDW**	EVE LK201

Related Commands

CHANGE WINDOW_MODE

DELETE WINDOW

ENLARGE WINDOW

ONE WINDOW

SET SCREEN

SHRINK WINDOW

TWO WINDOWS

QUIT

QUIT — Ends an LSE session without saving any modified user buffers.

Format

QUIT

Description

The QUIT command ends the editing session without saving modified user buffers.

If you have modified any buffers, LSE warns you that you have changes that will be lost and asks if you want to continue quitting. Typing Y or YES confirms that you want to discard the modified buffers; typing N or NO reactivates the editing session and returns the cursor to the last current buffer.

In DECwindows mode, if you have modified any buffers, LSE displays a dialog box to warn you that modifications will be discarded and to confirm that you want to continue quitting.

DECwindows Interface Equivalent

Pull-down menu: File → Quit

Related Commands

ATTACH

EXIT

SPAWN

QUOTE

QUOTE — Enters a control code or other character, either as text in the buffer you are editing or as a string for a command.

Format

QUOTE

Description

The QUOTE command enters the character according to the current mode of the buffer, as shown in the status line.

You can also use the QUOTE command for entering strings for search or substitute commands.

If you use the DEFINE KEY command to define a typing key (letter, number, or punctuation mark) or a control key, you can use the QUOTE command to enter the character or control code normally bound to that key.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl/V	EDT LK201, EDT VT100, EVE LK201

Related Commands

SET INSERT

SET OVERSTRIKE

Examples

To use the QUOTE command to enter strings for search or substitute commands, do the following:

1. Press the key defined for the SEARCH or SUBSTITUTE command.
2. Press Ctrl/V.
3. Press Ctrl/J for the line-feed character.

You can define a typing or control key, then use the QUOTE command to enter the character or control code normally bound to that key. For example,if you define the tilde to execute a procedure, insert a tilde character (~) by doing the following:

1. Press Ctrl/V.
2. Type the tilde.

READ

READ — Inserts the contents of a file into a buffer.

Format

READ file-spec

Qualifier	
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Specifies a buffer into which the file is to be read. If the buffer does not exist, it is created for display only (the buffer cannot be written back to a file).

Parameter

file-spec

Specifies the file to be read. LSE uses the list for the current SET SOURCE_DIRECTORY command to resolve the file specification.

LSE uses CMS to access a file, if the directory for the file to be accessed is the same as the current CMS library.

Description

The READ command opens a file for input and inserts the file's contents into a buffer. LSE inserts the text before the line containing the current position in the receiving buffer; if the buffer previously contained no text, the cursor is positioned at the end of the buffer.

Unless you specify otherwise, the receiving buffer is the current buffer.

Related Commands

GOTO FILE

INCLUDE

SET CMS

SET SOURCE_DIRECTORY

WRITE

Example

```
LSE> READ x.y
```

Opens file *x.y* for input and reads that file's contents into the current buffer.

RECALL

RECALL — Recalls a previous LSE command, which you can edit and execute again.

Format

RECALL

Description

The RECALL command recalls a previous LSE command, which you can edit (if necessary) and execute again. You cannot just enter RECALL to recall a previous LSE command. If you enter RECALL, the command itself is recalled. Instead, use GOLD/DO or a key defined as RECALL.

When you press GOLD/DO, the most recent command you entered is displayed in the command window with the cursor at the end of the command line. To execute the recalled command, press Return or the Do key.

To recall another command, press GOLD/DO again, or press the up arrow key (in effect, scrolling back through the command buffer.)

To cancel the recalled command, erase the recalled line (for example, by pressing Ctrl/U).

Do not enter the command RECALL. If you enter RECALL, that command itself is recalled. Instead, use GOLD/DO or a key defined as RECALL.

RECOVER BUFFER

RECOVER BUFFER — Reconstructs the contents of a buffer from a buffer-change journal file.

Format

RECOVER BUFFER [file-name]

Qualifier	Defaults
/ALL	

Qualifier

/ALL

Specifies that LSE should use the latest generation of all locatable buffer-change journal files to attempt to perform a recovery operation. LSE uses the file specification LSE$JOURNAL:.TPU $JOURNAL to locate all buffer-change journal files. If you specify the /ALL qualifier,you cannot specify the *file-name* parameter.

Parameter

file-name

Specifies the name of the file. You can specify either of the following files that the editor should use to perform the recovery operation:

- Source file that was in the buffer
- Full name of the buffer-change journal file

For information about the procedure for recovering changes lost in a system failure, see the section about recovering from a failed editing session in the *VSI DECset for OpenVMS Guide to Language-Sensitive Editor* and *VSI DECset for OpenVMS Guide to VSI Source Code Analyzer*.

Description

The RECOVER BUFFER command attempts to rebuild the contents of a buffer by using the latest available generation of the file that was in the buffer and a journal file that contains a description of the changes to that buffer. LSE uses the default file specification LSE$JOURNAL:*.TPU$JOURNAL when attempting to locate buffer-change journal files.

Before LSE attempts to recover a buffer, information about the journal file is displayed. When you specify the /ALL qualifier, LSE displays information about each available journal file in succession. You can choose not to recover a buffer if the information describes a journal file other than the one you want.

Related Commands

SET JOURNALING

SET NOJOURNALING

Example

```
LSE> RECOVER BUFFER login.com
```

Recovers the buffer LOGIN.COM from the journal file in LSE$JOURNAL:LOGIN_COM.TPU $JOURNAL.

REDO

REDO — Reverses an UNDO operation for the current buffer.

Format

REDO

Description

A follow-up command to UNDO, the REDO command reverses the UNDO command. A series of UNDO commands can be reversed by a series of REDO commands.

DECwindows Interface Equivalent

Pull-down menu: Edit → Redo

Related Commands

SET MAX_UNDO

SET MODE UNDO=OFF

SET MODE UNDO=ON

SHOW MAX_UNDO

UNDO

REFRESH

REFRESH — Refreshes the screen display.

Format

REFRESH

Description

The REFRESH command clears and redisplays the screen, which preserves all valid text,including messages in the message window. The cursor returns to its current position.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl/W	All

DECwindows Interface Equivalent

Pull-down menu: View → Refresh

Example

```
Ctrl/W
```

Causes the screen to go blank for a moment. The display then returns without any extraneous characters that do not belong in your displayed buffers.

REORGANIZE

REORGANIZE — Optimizes the organization of the specified SCA libraries.

Format

REORGANIZE [library-spec[, . . .]]

Qualifier	Defaults
/[NO]LOG	/LOG

Qualifier

/LOG (D)
/NOLOG

Indicates whether SCA reports a successful library reorganization.

Parameter

[library-spec[, . . .]]

Specifies the SCA libraries to be reorganized. If you do not specify a library, LSE reorganizes the primary SCA library.

Description

The REORGANIZE command optimizes the organization of SCA libraries soyou get the best query and update performance.

Example

```
$ SCA
SCA> CREATE LIBRARY library-directory /MODULE_COUNT=...
SCA> LOAD data-file-directory:*.ANA
SCA> REORGANIZE
```

Creates and optimizes the size and organization of your SCA library.

REPEAT

REPEAT — Repeats a command the specified number of times.

Format

REPEAT repeat-count command

Parameters

repeat-count

Specifies a positive decimal integer number indicating the number of times you want to repeat the command.

command

Specifies the command to be repeated.

Description

The REPEAT command repeats a command the number of times you specify.

To repeat a single key, press the PF1 key followed by one or more keyboard number keys to indicate the number of times you want the key to be repeated. Then, press the key you want.

You cannot use the PF1 key to repeat the delete key or Ctrl/Z key.

The repeat operation aborts if you receive a warning of an error while this command is active.

Keypad Equivalent

Key	Keypad Mode
PF1-number key(s)	All

Examples

1. `LSE>` ***`REPEAT 5 ENTER LINE`***

   Adds five new lines to the text in the current buffer.

2. ***`PF1 7 0 =`***

   Inserts 70 equals signs (=) at the current cursor position.

REPLACE

REPLACE — Creates a new generation of the specified element in your current CMS library.

Format

REPLACE

Qualifier	Default
/[NO]VARIANT=variant-letter	/NOVARIANT

Qualifier

/VARIANT=variant-letter
/NOVARIANT (D)

Controls whether CMS creates a variant generation.

Description

The REPLACE command returns to your current CMS library an element name with the same name and type as the input file for your current buffer. When a REPLACE command executes successfully, it creates a new generation of that element; you no longer hold a reservation for the element.

The sequence of actions this command takes are as follows:

1. Writes out the buffer if you have modified it.
2. Performs a CMS REPLACE operation.
3. Deletes the buffer.

DECwindows Interface Equivalent

Pull-down menu: File → Replace

Related Commands

RESERVE

SET CMS

UNRESERVE

Example

```
LSE> REPLACE
```

Creates a new generation of the element with the same name and type as the input file for your current buffer.

REPORT

REPORT — Produces the specified report.

Format

REPORT report-name other-parms[. . .]

Qualifiers	Defaults
/DOMAIN=query-name	
/[NO]FILL	/FILL
/HELP_LIBRARY=library_name	
/LANGUAGES=(language,[. . .])	
/OUTPUT=file-name	
/TARGET=target-file-type	See text

Qualifiers

/DOMAIN=query-name

Specifies the name of the query to use as the domain for the report. The query should include occurrences of files that have been compiled. This value is converted to a DECTPU value and assigned to the global DECTPU variable SCA$REPORT_DOMAIN_QUERY. This procedure limits the report to objects that are contained, directly or indirectly,within at least one of the files in this query.

The default value is the null string. By convention, DECTPU report procedures interpret this as the entire SCA library.

/FILL (D)
/NOFILL

Specifies that whenever a paragraph of commented text is inserted into a report, it is set up so a text processor, such as DECdocument, performs the usual fill and justification operations on the paragraph. If you specify /NOFILL, there port tool does not instruct the text processor to fill or justify the paragraph.

For any individual paragraph, you can override the setting of this qualifier by including appropriate text-processor comments within the body of the comment.

The value of this qualifier is used to set the value of the global DECTPU variable SCA $REPORT_FILL as follows. If you specify the /FILL qualifier,or it is specified by default, SCA $REPORT_FILL is 1; if you specify the /NOFILL qualifier, SCA$REPORT_FILL is 0.

The /FILL qualifier is ignored if it is not meaningful for the target. In particular, it is ignored for LSE package definitions.

/HELP_LIBRARY=library_name

Specifies the help library to use for PACKAGE reports. This qualifier is ignored for other reports. The PACKAGE report generates one or more DEFINE PACKAGE commands. The *library_name* specifies the value to use with the /HELP_LIBRARY qualifier for the generated DEFINE PACKAGE commands.

If you omit this qualifier, the PACKAGE report omits the /HELP_LIBRARY qualifier from the DEFINE PACKAGE commands it generates.

/LANGUAGES=(language,[. . .])

Specifies the language to use for PACKAGE reports. This qualifier is ignored for other reports. The PACKAGE report generates one or more DEFINE PACKAGES commands. This qualifier specifies the languages to use as the values of the /LANGUAGE qualifier for the generated DEFINE PACKAGE commands.

If you omit this qualifier, the PACKAGE report inserts the LSE placeholder *{language_name}* … as the value for the /LANGUAGE qualifier with the DEFINE PACKAGE commands. Before you can execute the DEFINE PACKAGE command, you must replace the placeholder manually with the names of the languages that are appropriate for the languages being defined.

/OUTPUT=file-name

Specifies the output file to use for the report. This value is converted to a DECTPU string and passed as the value for the global DECTPU variable SCA$REPORT_OUTPUT. The default value takes the file name from the *report-name* parameter and the file type from the *target-file-type* parameter. The *target-file-type*is implied by the /TARGET qualifier. For example, if you specify DECdocument for the /TARGET qualifier, this implies a file type of .SDML.

/TARGET=target-file-type

Specifies the type of target file to produce. This value is converted to a DECTPU string value and assigned to the global DECTPU variable SCA$REPORT_TARGET. You can specify one of the following keywords:

Keyword	Type of file
TEXT, TXT	Text file
RUNOFF, DSR, RNO	A file for processing by DIGITAL RUNOFF
SDML, DOCUMENT	A file for processing by DECdocument
LSEDIT, LSE	A file for processing by LSE
HLP, HELP	A help file for processing by the VMS Librarian
OTHER=value [1]	Optional file type

[1]The OTHER keyword can take an optional value. The default value is the null string, which by convention is interpreted by the DECTPU procedures as TEXT. User-supplied report procedures can ignore this convention and provide their own defaults.

The default target file types are SDML for INTERNALS and 2167A_DESIGNreports, HLP for HELP reports, and LSE for PACKAGE reports.

Parameters

report-name

Specifies the name of the report to produce. The command looks for a corresponding DECTPU procedure by constructing the DECTPU identifier SCA_REPORT_ *report-name* and looking for a DECTPU procedure with that name to use for producing the report. Because DECTPU limits identifiers to 132 characters, report names are limited to 132 minus LENGTH “SCA_REPORT_”, which equals 121 characters.

VSI has implemented the following reports:

- **HELP—**A help file suitable for processing by the VMS Librarian into a help library
- **PACKAGE—**An LSE package definition, which can be processed by LSE and put into an environment file, to create templates for calling the procedures in your code
- **INTERNALS—**A comprehensive report on the software in your system,all of the information in comment headers, and a structural presentation of your code
- **2167A_DESIGN—**The design section of the DOD-STD-2167A Software Design Document

You must type report names completely as they appear in the previous list.

other-parms[. . .]

Specifies other parameters passed to the DECTPU procedure. These parameters are collected into a single string, which is then assigned to the global DECTPU variable SCA$_REPORT_REST_OF_LINE. These SCA parameters are obtained from the command line from the $REST_OF_LINE type of the OpenVMS Command Definition Utility. For information on the built-in value type $REST_OF_LINE, see the section about defining values in the *VSI OpenVMS Command Definition, Librarian, and Message Utilities Manual*.

Description

The REPORT command produces a specified report. For more information about the REPORT command and about customizing reports, see the chapter about customizing reports in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

The REPORT command requires that LSE be installed, even if you are using this command from the SCA command line.

Example

1. `LSE>` ***`REPORT HELP /TARGET=HELP`***

   Produces a report named HELP with a file type of .HLP for processing with the VMS Librarian.

2. `SCA>` ***`FIND/NAME=abc_files abc* AND symbol=file AND occ=command_line`***
   `SCA>` ***`REPORT/DOMAIN=abc_files INTERNALS`***

   Produces an INTERNALS report only on files with names beginning with *abc*.

RESERVE

RESERVE — Reserves an element in your current CMS library.

Format

`RESERVE [element-name]`

Qualifiers	Default
/GENERATION[=generation-exp]	
/[NO]MERGE=generation-exp	/NOMERGE

Qualifiers

/GENERATION[=generation-exp]

Specifies the generation of the element to reserve. If you do not specify a value, LSE assumes you have specified generation “1+” (the *generation-exp* parameter must be enclosed in quotation marks if non-alphanumeric characters are present). If you omit the qualifier altogether, LSE uses the specified or default value from the command SET CMS/GENERATION to determine the generation to reserve.

/MERGE=generation-exp
/NOMERGE (D)

Determines whether LSE merges another generation of the element with the generation being reserved. If you omit this qualifier, LSE uses the setting of the command SET CMS/[NO]MERGE to determine whether to merge generations of the element being reserved.

You must enclose *generation-exp* in quotes (“ ”) if non-alphanumeric characters are present.

Parameter

element-name

Specifies the elements to reserve. If you do not specify an element name, LSE uses the file name and type of your current buffer as the element name.

Description

The RESERVE command executes the CMS command RESERVE on the specified element in your current CMS library and reads the file created into the current editing buffer.

To specify conditions for reserving the element, use the SET CMS command with its available command qualifiers.

DECwindows Interface Equivalent

Pull-down menu: File → Reserve

Related Commands

REPLACE

SET CMS

UNRESERVE

Example

```
LSE> RESERVE USER.TXT
```

Reserves an element called USER.TXT in your current CMS library.

REVIEW

REVIEW — Selects and displays a set of diagnostic messages that resulted from a compilation. The diagnostics associated with the current contents of the buffer become the current diagnostic set.

Format

REVIEW [buffer]

Qualifier	Defaults
/FILE=file-spec	

Qualifier

/FILE=file-spec

Specifies the name of the diagnostics file containing the results of a compilation. By default, LSE looks in your current directory for a .DIA file with the same file name as the file associated with the buffer.

Parameter

buffer

Specifies that the set of diagnostics associated with the specified buffer is to be reviewed. The default is the current buffer.

Description

The REVIEW command selects and displays a set of diagnostic messages associated with the current contents of a buffer.

A set of diagnostics becomes associated with a buffer by a COMPILE/REVIEW or REVIEW command. It remains associated with that buffer until you enter a subsequent COMPILE command for that buffer, a REVIEW command with an explicit /FILE qualifier, or an END REVIEW command.

If no diagnostics are associated with the buffer, LSE attempts to read a set of diagnostics from a file. If you do not supply a file specification, LSE uses the name of the file associated with the buffer, but with .DIA as the file type. You can use the /FILE qualifier to override this default.

You can use the REVIEW command at any time to change the set of diagnostics to be reviewed. If you use the REVIEW command to return to a set of diagnostics,the last diagnostic and region selected in that set become the current diagnostic and region.

DECwindows Interface Equivalent

Pull-down menu: Source → Review

Related Commands

COMPILE/REVIEW

END REVIEW

GOTO REVIEW

Example

```
LSE> REVIEW
```

Displays compilation diagnostics in a window containing the $REVIEW buffer,after you have used the /DIAGNOSTICS qualifier to invoke a compiler.

SAVE ENVIRONMENT

SAVE ENVIRONMENT — Writes out all user-defined languages, placeholders, tokens, aliases, and packages to an environment file.

Format

SAVE ENVIRONMENT file-spec

Qualifiers	Defaults
/ALL	/ALL

Qualifiers	Defaults
/NEW	/ALL

Qualifiers

/ALL (D)

Specifies that LSE write all defined items to the environment file.

/NEW

Specifies that LSE write out only those definitions you made during the current editing session. Definitions that were read in from an environment file are not written.

Parameter

file-spec

Specifies the file to which LSE should write the environment data.

Description

The SAVE ENVIRONMENT command writes out all user-defined languages,placeholders, tokens, aliases, and packages to an environment file. This procedure saves processing time when LSE reads the definitions back in. (See the section about using environment and section files in the *Guide to Language-Sensitive Editor for OpenVMS Systems* for information on the use of the logical name LSE $ENVIRONMENT or the LSE command-line qualifier/ENVIRONMENT to restore definitions in an environment file.)

Usually, LSE writes all user-defined items to the environment file. You can supply user-defined items with the LSEDIT /INITIALIZATION and /ENVIRONMENT qualifiers, or with DEFINE commands during the editing session. You can use the /NEW qualifier to tell LSE to write only those items defined during the current editing session.

DECwindows Interface Equivalent

Pull-down menu: Options → Save Options

Related Commands

`SAVE SECTION`

Example

```
LSE> SAVE ENVIRONMENT myfile.env
```

Creates an environment file named myfile.env to hold any current language, placeholder, token, alias, and package definitions.

SAVE QUERY

SAVE QUERY — Saves queries from the SCA query list into a command file.

Format

SAVE QUERY [query-name, . . .]

Qualifiers	Defaults
/OUTPUT[=file-spec]	QUERY.COM
/PREFIX=name-prefix	
/QUALIFIERS=find-command-qualifiers	

Qualifiers

/OUTPUT[=file-spec]

Specifies an output file name and overrides the default QUERY.COM.

/PREFIX=name-prefix

Adds the specified prefix to all query names. This qualifier can be used to make sure the query names are unique, and is useful if any of the query names are numbers.

/QUALIFIERS=find-command-qualifiers

Used to specify FIND command qualifiers to be added to each saved query. For example, the value of "/NORESULT"will prevent the queries from being evaluated when they are read into SCA until they are used.

Parameter

query-name

Specifies the name of an existing command file to receive the queries from the SCA query list.

Description

The SAVE QUERY command saves queries from an SCA query list into a command file. The saved query can then be read into any SCA session by using the @file-specification command.

Related Commands

@ (file-specification)

FIND

Example

```
$ SCA
SCA> FIND/NAME=Q1/NORESULT NAME11 OR NAME12
SCA> FIND/NAME=Q2/NORESULT NAME21 OR NAME22
SCA> FIND/NAME=Q/NORESULT @Q1 OR @Q2
SCA> SAVE QUERY/PREFIX=X_/QUALIFIERS="/NORESULT"
SCA> EXIT
$ TY QUERY.COM
FIND/NAME=X_Q1/NORESULT NAME11 OR NAME12
```

```
FIND/NAME=X_Q2/NORESULT NAME21 OR NAME22
FIND/NAME=X_Q/NORESULT @X_Q1 OR @X_Q2
```

This example demonstrates the use of the SAVE QUERY command in OpenVMS syntax format.

Note

A query for which there are no matches will not be put in the query list unless it is defined with the /NORESULT qualifier.

SAVE SECTION

SAVE SECTION — Writes the binary form of all current key definitions, learn sequences, and DECTPU procedures and variables to a section file. This saves processing time when LSE reads the definitions back in.

Format

SAVE SECTION file-spec

Qualifiers	Defaults
/[NO]DEBUG_NAMES	/DEBUG_NAMES
/IDENT=string	
/[NO]PROCEDURE_NAMES	/PROCEDURE_NAMES

Qualifiers

/DEBUG_NAMES (D)
/NODEBUG_NAMES

Specifies whether DECTPU procedure parameters or local variable names should be written to the section file.

/IDENT=string

Specifies an identifying string for the section file.

/PROCEDURE_NAMES (D)
/NOPROCEDURE_NAMES

Specifies whether DECTPU procedure names should be written to the section file.

Parameter

file-spec

Specifies the file to which LSE should write the section data. The default file type is .TPU$SECTION.

Description

The SAVE SECTION command writes key definitions, learn sequences,user-defined commands, mode settings, DECTPU procedures, and DECTPU variable names to a section file so they can be restored at a

later time. (See the section about using environment and section files in the *Guide to Language-Sensitive Editor for OpenVMS Systems* for information on the use of the logical name LSE$SECTION or the LSE command line qualifier /SECTION, to restore definitions saved in the section file.)

The SAVE SECTION command calls the DECTPU built-in SAVE procedure to actually write the section file. By default, the type of the saved section file is .TPU$SECTION.

DECwindows Interface Equivalent

Pull-down menu: Options > Save Options …

Related Commands

SAVE ENVIRONMENT

Example

```
LSE> SAVE SECTION MY_SECTION
```

Creates a section file named MY_SECTION.TPU$SECTION; the file saves all current key definitions, learn sequences, DECTPU procedures, and variable names.

SEARCH

SEARCH — Searches the current buffer for the specified string and positions the cursor at that string.

Format

SEARCH search-string

Qualifiers	Defaults
/DIALOG	/NODIALOG
/[NO]PATTERN	/NOPATTERN

Qualifiers

/DIALOG
/NODIALOG (D)

Instructs LSE to use a dialog box to prompt the user for parameters and qualifier values. The command parameters are optional if you specify this qualifier. If you supply command parameters and qualifiers with the /DIALOG qualifier,these parameters and qualifiers are used to set the initial state of the dialog box.

LSE ignores the /DIALOG qualifier if you are using a character-cell terminal.

/PATTERN
/NOPATTERN (D)

Enables or disables special interpretation of wildcard characters and a quote character in the *search-string* parameter. You can set the syntax for specifying a pattern to the OpenVMS style (/PATTERN=OPENVMS), UNIX style (/PATTERN=ULTRIX) or TPU style (/PATTERN=TPU).

For more details on TPU patterns see Appendix G and *DEC Text Processing Utility Reference Manual*. *Table 2.5, "OpenVMS Wildcards"*lists the OpenVMS wildcards. *Table 2.6, "UNIX Wildcards"* lists the UNIX wildcards.

Table 2.5. OpenVMS Wildcards

Wildcard	Matches
*	One or more characters of any kind on a line.
**	One or more characters of any kind crossing lines.
%	A single character.
\ <	Beginning of a line.
\>	End of a line.
\[set-of-characters]	Any character in the specified set. For example, \[abc] matches any letter in the set “abc” and \[c-t] matches any letter in the set “c” through “t.”
\[~set-of-characters]	Anything not in the specified set of characters.
\	Lets you specify the characters \,*,% or]within wildcard expressions. For example, \ \ matches the backslash character (\).
\.	Repeats the previous pattern zero or more times,including the original.
\:	Repeats the previous pattern at least once, including the original; that is, a null occurrence does not match.
\w	Any empty space created by the space bar or tab stops,including no more than one line break.
\d	Any decimal digit.
\o	Any octal digit.
\x	Any hexadecimal digit.
\a	Any alphabetic character, including accented letters,other marked letters, and non-English letters.
\n	Any alphanumeric character.
\s	Any character that can be used in a symbol:alphanumeric, dollar sign, and underscore.
\l	Any lowercase letter.
\u	Any uppercase letter.
\p	Any punctuation character.
\f	Any formatting characters: backspace, tab, line feed,vertical tab, form feed, and carriage return.
\^	Any control character.
\+	Any character with bit 7 set; that is, ASCII decimal values from 128 through 255.

Table 2.6. UNIX Wildcards

Wildcard	Matches
.	A single character.
^	Beginning of a line.
$	End of a line.
[set-of-characters]	Any character in the specified set. For example, [abc] matches any letter in the set “abc” and[c-t] matches any letter in the set “c” through “t.”
[^set-of-characters]	Anything not in the specified set of characters.
\	Lets you specify the characters \,.,^,$,[,],or* in wildcard expressions. For example, \ \ matches the backslash character (\).
*	Repeats the previous pattern zero or more times,including the original.
+	Repeats the previous pattern at least once, including the original; that is, a null occurrence does not match.

When you specify the /NOPATTERN qualifier (or when it is the default), special interpretation of the asterisk, percent sign, and backslash characters is disabled.

Parameter

search-string

Specifies a quoted string indicating the string for which to search.

If you are using the DECwindows interface and specify the /DIALOG qualifier, the search string field in the Find dialog box takes the default value from the previous search string, if any.

Description

The SEARCH command searches the current buffer in the specified direction for the specified character string, but ignores any occurrence of the search string that begins at the current cursor position. If the search is successful, LSE positions the cursor on the first character of the string. If LSE does not find the string, it issues a message indicating that no matching string was found.

The direction in which a search is performed is independent of the current direction set for a buffer. This lets you change the direction of the search operation without changing the current direction set for the buffer. The prompts for the search string reflect this behavior. Note that you can change the direction of the search by pressing a key that changes the search direction;this can be the first key you press in response to the prompt, or the key that terminates the prompt.

When conducting a search, LSE regards uppercase and lowercase letters as equivalent. To alter this behavior, see the SET SEARCH command.

If you specify a null string as the search string, LSE searches for the last search string given in the SEARCH command. If LSE prompts you for a search string, you must not use quotation marks in your response unless you want LSE to search for a string that includes quotation marks.

The direction in which LSE executes the SEARCH command is determined by the key used to end the SEARCH command. If you end your response to the prompt with a keypad key bound to SET FORWARD or SET REVERSE, LSE changes the search direction before the SEARCH command. This is not true in DECwindows if you are specifying search strings through the dialog box.

Keys bound to other commands end the string and LSE conducts the search in the current direction.

For information about searching for a formatting or control character,see the QUOTE command.

Keypad Equivalent

SEARCH

Key	Keypad Mode
PF1-PF3 **FIND**	EDT LK201, EDT VT100, EVE LK201
E1 **FIND**	EDT LK201, EVE LK201
KP4 **FIND**	EVE VT100

SEARCH " "

Key	Keypad Mode
PF3 **FNDNXT**	EDT LK201, EDT VT100, EVE LK201

SEARCH/PATTERN

Key	Keypad Mode
PF1-E1 **FNDPATT**	EDT LK201, EVE LK201

DECwindows Interface Equivalent

SEARCH/DIALOG

Pull-down menu: Search → Search . . .

SEARCH ""

Pop-up menu: User buffer → Find Next

Pull-down menu: Search → Search . . .

Related Commands

QUOTE

SET SEARCH

SHOW SEARCH

Examples

1. `LSE> SEARCH "the editor"`

Searches the current buffer for the next occurrence of the string *the editor*. The quotation marks in the search string indicate to LSE that you are searching for the words enclosed in the quotation marks.

2. ***FIND***
   _Forward Search: ***open***

   Searches the current buffer for the next occurrence of the word *open*.

3. LSE> ***SEARCH/PATTERN "2%\%"***

   Searches the current buffer for the next occurrence of a string consisting of the number *2*, any character, and a percent sign. Text that would satisfy this condition includes the strings "20% " and "29%. "

SELECT ALL

SELECT ALL — Selects the entire contents of the current buffer.

Format

SELECT ALL

Description

The SELECT ALL command places all the contents of the current buffer in the selected range. Any operations that LSE performs on a selected range then apply to all the contents of the buffer.

DECwindows Interface Equivalent

Pull-down menu: Edit → Select All

SET AUTO_ERASE

SET AUTO_ERASE — Enables automatic erasing of placeholders in the specified buffer.

Format

SET AUTO_ERASE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer for which automatic erasing is to be enabled. The default is the current buffer.

Description

The SET AUTO_ERASE command enables LSE to erase the placeholder that the cursor is on when you type a character over that placeholder in the specified buffer. However, if the cursor is on the first character of an open placeholder delimiter, LSE displays the characters you type without erasing the placeholder.

Initially, LSE is set to automatically erase placeholders.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

SET NOAUTO_ERASE

SHOW BUFFER

Example

```
LSE> SET AUTOERASE/BUFFER=USER.TXT
```

Enables automatic erasing of placeholders in the buffer USER.TXT.

SET CMS

SET CMS — Sets the default values for reservations and fetches that LSE performs when you enter the appropriate LSE file-manipulation commands.

Format

SET CMS

Qualifiers	Defaults
/[NO]CONCURRENT	
/[NO]CONFIRM	
/GENERATION=generation-exp	
/[NO]HISTORY	
/[NO]MERGE=generation-exp	
/[NO]NOTES	
/[NO]REMARK=string	

Qualifiers

/CONCURRENT
/NOCONCURRENT

Controls whether an element reserved by you can be reserved by another user while you have it reserved. The initial setting is /CONCURRENT.

/CONFIRM
/NOCONFIRM

Specifies whether you want to be prompted for confirmation before LSE performs a FETCH or RESERVE operation. The initial setting is /CONFIRM.

/GENERATION=generation-exp

Specifies the generation to be used for CMS RESERVE and FETCH operations. The initial setting is /GENERATION= “1+”.

/HISTORY
/NOHISTORY

Controls whether CMS includes the element history in the file if the element has the history attribute and if a CMS FETCH or CMS RESERVE operation is performed. The initial setting is /HISTORY.

/MERGE=generation-exp
/NOMERGE

Controls whether LSE merges a reserved or fetched element with another generation of the same element. The initial setting is /NOMERGE.

/NOTES
/NONOTES

Controls whether notes are embedded in the file if the retrieved element has the notes attribute and if a CMS FETCH or CMSRESERVE operation is performed. The initial setting is /NOTES.

/REMARK=string
/NOREMARK

Specifies the remark to be used on RESERVE operations. The initial setting is to prompt for the remark. If you specify the /NOREMARK qualifier, LSE prompts you for a remark when you enter a CMS file-manipulation command.

Description

The SET CMS command specifies default settings for the LSE file-manipulation commands that reserve or fetch files.

The effect of the SET CMS command is cumulative; that is, entering a SETCMS/NOHISTORY command followed by a SET CMS/NONOTES command causes both/NOHISTORY and /NONOTES to be set. (You would then need to enter the command SETCMS/HISTORY to set /HISTORY again.)

If you do not specify any qualifiers, the SET CMS command resets all values to their initial settings.

Note

The SET CMS command settings are not used by any commands that begin with the word CMS.

DECwindows Interface Equivalent

Pull-down menu: Options → CMS . . .

Related Commands

GOTO FILE

GOTO SOURCE

READ

REPLACE

RESERVE

SHOW CMS

UNRESERVE

Example

```
LSE> SET CMS/GENERATION=Baselevel_1
```

Causes fetches performed by the commands GOTO FILE, GOTO SOURCE, and READ to use the generation that corresponds to the class *Baselevel_1*. Any reservations made using the RESERVE command also use this class.

SET CURSOR

SET CURSOR — Selects either bound cursor motion or free cursor motion.

Format

SET CURSOR motion-setting

Parameter

motion-setting

Specifies the cursor-motion setting. Motion-setting keywords and their effects are as follows:

BOUND

Restricts the cursor to positioning on a character, end-of-line, or end-of-buffer. This is the initial setting and is similar to cursor motion in the EDT editor.

FREE

Lets the cursor move anywhere in a window including past the end-of-line, past the end-of-buffer, in the middle of a tab, or to the left of the left margin. This is similar to the default cursor motion for the EVE editor.

Description

The SET CURSOR command either binds the cursor to that part of the buffer occupied by text, or sets it free to be positioned anywhere in the buffer,depending on the parameter you specify.

DECwindows Interface Equivalent

Pull-down menu: Customize → Global Attributes . . .

Related Commands

SHOW MODE

SET DEFAULT_DIRECTORY

SET DEFAULT_DIRECTORY — Changes your default device and directory specifications.

Format

SET DEFAULT_DIRECTORY [device-name[:][directory-spec]

Parameters

device-name[:]

Specifies a device name to be used as the default device in a file specification.

directory-spec

Specifies a directory name to be used as the default directory in a file specification. A directory name must be enclosed in brackets. Use the minus sign to specify the next higher directory from the current default directory.

You must specify either the *device-name* parameter or the *directory-spec* parameter. If you specify only the device name, the current directory is the default for the *directory-spec* parameter. If you specify only the directory name, the current device is the default for the *device-name* parameter.

You can use a logical name, but it must constitute at least the device part of the specification.

Description

The SET DEFAULT_DIRECTORY command changes your default device and directory names, along with any equivalence strings. The new default is applied to all subsequent file specifications that do not explicitly include a device or directory name.

The default set in an LSE editing session remains in effect after you terminate the LSE session.

Related Commands

SHOW DEFAULT_DIRECTORY

Example

```
LSE> SET DEFAULT DISK$:[USER.LSE]
```

Establishes DISK$:[USER.LSE] as the default directory for LSE to use in accessing files.

SET DIRECTORY

SET DIRECTORY — Sets the default read-only or writable status of files in a specified directory.

Format

SET DIRECTORY directory-spec

Qualifiers	Defaults
/READ_ONLY	/WRITE
/WRITE	/WRITE

Qualifiers

/READ_ONLY

Specifies that files in the specified directories are read-only and unmodifiable by default. The /READ_ONLY qualifier prevents the WRITE command from writing files to the specified directory, unless you subsequently override this default.

/WRITE (D)

Specifies that files in the specified directories are writable and unmodifiable by default.

Parameter

directory-spec

Specifies a directory to be set as read-only or writable.

Description

The SET DIRECTORY command determines the read-only or writable status of a directory you specify. The logical name LSE$READ_ONLY_DIRECTORY stores the list of read-only directories.

Related Commands

SHOW DIRECTORY

Example

```
LSE> SET DIRECTORY/READ_ONLY [LIBRARY_DIRECTORY]
```

Specifies files in the directory LIBRARY_DIRECTORY as unmodifiable.

SET FONT

SET FONT — Sets the specified fonts for the screen.

Format

SET FONT keyword-list

Parameter

keyword-list

Indicates the fonts to be set or reset. The types of fonts areas follows:

BIG

Specifies that the fonts should be big

CONDENSED

Specifies that the fonts should be condensed

LITTLE

Specifies that the fonts should be little

NORMAL

Specifies that the fonts should be normal

Description

The SET FONT command sets the fonts to big or little,normal or condensed. You can specify either big or little,and either normal or condensed.

You use the SET FONT command only with DECwindows.

DECwindows Interface Equivalent

Pull-down menu: Customize > Window Attributes . . .

Related Commands

SHOW SCREEN

Example

```
LSE> SET FONT BIG,CONDENSED
```

Set the fonts to big and condensed.

SET FORWARD

SET FORWARD — Sets the current direction of a buffer to forward.

Format

SET FORWARD

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose direction is to be set to forward. The default is the current buffer.

Description

The SET FORWARD command sets the current direction of the specified buffer to forward. The status line of each buffer displays the current direction.

Users of the DECwindows interface can switch direction by selecting the status line button and pressing MB1.

Keypad Equivalent

Key	Keypad Equivalent
KP4 **FORWARD**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

DECwindows Interface Equivalent

Buffer status line $\left\{\begin{matrix}\text{Forward}\\\text{Reverse}\end{matrix}\right\}$

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

CHANGE DIRECTION

SET REVERSE

SET INDENTATION

SET INDENTATION — Sets the current indentation-level count for the current buffer without changing the current line.

Format

SET INDENTATION level-option

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose current indentation-level count is to be changed. The default is the current buffer.

Parameter

level-option

Indicates the level to be set or changed. The indentation keywords and their effects are as follows:

CURRENT

Sets the indentation level count to the beginning of the text on the current line

CURSOR

Sets the indentation level count to the column currently occupied by the cursor

LEFT

Decreases the indentation level count by the current tab increment

RIGHT

Increases the indentation level count by the current tab increment

Description

The SET INDENTATION command sets the current indentation-level count for the current buffer. A TAB or ENTER TAB command given at the beginning of a line inserts tabs and blanks corresponding to the current indentation-level count.

DECwindows Interface Equivalent

Pull-down menu: Edit → Indentation . . .

Related Commands

CHANGE INDENTATION

ENTER TAB

EXPAND

TAB

UNTAB

SET INSERT

SET INSERT — Sets the text-entry mode of the specified buffer to insert mode.

Format

SET INSERT

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose text-entry mode is to be changed. The default is the current buffer.

Description

The SET INSERT command sets the mode of the specified buffer to insert. In insert mode, LSE inserts typed characters before the current cursor position.

The status line of each buffer displays the current text-entry mode.

Users of the DECwindows interface can cycle through Insert, Overstrike, and Unmodifiable by selecting the status line button and pressing MB1.

DECwindows Interface Equivalent

Buffer status line {Insert / Overstrike / Unmodifiable}

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

CHANGE TEXT_ENTRY_MODE

SET OVERSTRIKE

SET JOURNALING

SET JOURNALING — Enables buffer-change journaling for the specified buffers.

Format

SET JOURNALING [buffer-name]

Qualifier	Defaults
/ALL	

Qualifier

/ALL

Specifies that all of the LSE user buffers that exist when the command is entered should be journaled. If you specify the /ALL qualifier, you cannot specify the *buffer-name* parameter.

Parameter

buffer-name

Specifies the name of the buffer that should be journaled. If you omit this parameter, the default is the current buffer.

Description

The SET JOURNALING command starts buffer-change journaling for the specified user buffer. SET JOURNALING does not allow buffer-change journaling for system buffers.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

RECOVER BUFFER

SET NOJOURNALING

Example

```
LSE> SET JOURNALING login.com
```

Enables buffer-change journaling for the buffer *login.com*.Buffer changes are written to the file LSE $JOURNAL:LOGIN_COM.TPU$JOURNAL.

SET LANGUAGE

SET LANGUAGE — Sets the language associated with the specified buffer.

Format

SET LANGUAGE language-name

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose associated language you want to set. The current buffer is the default.

Parameter

language-name

Specifies the name of the language to associate with the buffer. The SETLANGUAGE command requires this parameter.

Description

The SET LANGUAGE command associates a language with a buffer. By default, LSE uses a file-type specification to determine the language to associate with the buffer. If LSE cannot determine the language from the file type, or if no file is associated with the buffer, LSE uses the language in effect when you created the buffer. If you attempt to associate a language with a system buffer, such as $REVIEW,$MESSAGES, or $HELP, you receive an error message.

To disassociate a language with a specified buffer, use the SET NOLANGUAGE command.

DECwindows Interface Equivalent

Pull-down menu:Options → Buffer Attributes . . .

Related Commands

DEFINE LANGUAGE

DELETE LANGUAGE

SET NOLANGUAGE

SHOW LANGUAGE

Example

```
LSE> SET LANGUAGE example
```

Associates the language *example* with the current buffer.

SET LEFT_MARGIN

SET LEFT_MARGIN — Sets the left margin for the specified buffer.

Format

SET LEFT_MARGIN column-number

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose left margin is to be changed. The default is the current buffer.

Parameter

[column-number]

Specifies the column for the left margin. The value must be greater than or equal to 1, and less than the value set for the right margin.

If you specify the CONTEXT_DEPENDENT value as the column number, LSE uses the indentation of the current line to determine the left margin when you use the /WRAP qualifier. When you use the FILL command, LSE uses the indentation of the first line of each selected paragraph to determine the left margin.

Description

The SET LEFT_MARGIN command sets the left margin for a buffer. The FILL and ENTER LINE commands use this margin setting. By default, the left margin is at column 1.

To find out the setting of the left margin, use the SHOW BUFFER command.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

SET RIGHT_MARGIN

SHOW BUFFER [DECwindows interface]

Example

```
LSE> SET LEFT_MARGIN 10
```

Sets the left margin in the current buffer at column 10.

SET LIBRARY

SET LIBRARY — Identifies the SCA physical libraries to be used for subsequent SCA functions.

Format

SET LIBRARY directory-spec[, . . .]

Qualifiers	Default
/AFTER[=library-spec]	
/BEFORE[=library-spec]	
/[NO]LOG	/LOG

Qualifiers

/AFTER[=library-spec]

Instructs SCA to insert the new library or libraries into the list of active SCA libraries after the library you specify as the qualifier value. If you do not specify a value, SCA adds the library or libraries to the end of the list.

/BEFORE[=library-spec]

Instructs SCA to insert the new library or libraries into the list of active SCA libraries before the library you specify as the qualifier value. If you do not specify a value, SCA adds the library or libraries to the beginning of the list.

/LOG (D)
/NOLOG

Indicates whether SCA reports the resulting list of active SCA libraries.

Parameter

directory-spec[, . . .]

Specifies one or more directories, each of which comprises a separate SCA library. The list of libraries you specify replaces the current list of active libraries, unless you specify an /AFTER or /BEFORE qualifier.

Description

The SET LIBRARY command lets you activate the specified library for use during the current SCA session. If you list several directories, SCA can access all of them during your session as a single logical library. When you subsequently invoke SCA, it uses the logical name SCA$LIBRARY to reestablish the active library list.

Related Commands

`SET NOLIBRARY`

Example

```
$ SCA SET LIBRARY DISK$:[USER.SCALIB]
```

Defines the library named as the one SCA uses for subsequent access.

See the chapter about using SCA libraries in the *VSI DECset for OpenVMS Guide to Source Code Analyzer* for additional examples.

SET MARK

SET MARK — Associates a marker name with the current cursor position. You can later use that marker name with the GOTO MARK command to return to the specified position.

Format

`SET MARK marker-name`

Parameter

marker-name

Specifies the name of the marker to be placed. For a marker name, you can use any combination of up to 21 alphanumeric characters, underscores, or dollar signs. If this marker name is already in use, the previous marker is canceled.

Description

The SET MARK command tells LSE to remember the current cursor position by a marker. The command is useful if you are editing a large file and want to go back to a particular point in the text without having to search through the file.

Example

```
LSE> SET MARK M
```

Sets a marker named *M* as the reference for the current cursor position. Thereafter, entering the command GOTO MARK M returns the cursor to this position.

SET MAX_UNDO

SET MAX_UNDO — Sets the maximum number of UNDO operations that you can perform for a specific buffer.

Format

SET MAX_UNDO [/BUFFER=buffer-name] undo-number

Qualifiers	Defaults
/BUFFER=buffer-name	See text

Qualifiers

/BUFFER=buffer-name

Indicates the buffer whose maximum undo number is to be changed. The default is the current buffer.

Parameter

undo-number

Indicates the maximum undo number.

Description

Sets the maximum number of UNDO operations you can perform fora specific buffer. The default maximum value is 100.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

REDO

SET MODE UNDO=OFF

SET MODE UNDO=ON

SHOW MAX_UNDO

UNDO

Example

```
LSE> SET MAX_UNDO 60
```

Sets the maximum number of UNDO operations to 60.

SET MODE

SET MODE — Establishes the status of warning bells sounding, keypad emulation, select range, UNDO processing, tab appearance, tab characters, and the use of graphic characters in menus.

Format

SET MODE keyword-list

Parameter

keyword-list

Indicates the modes to be set or reset. The mode keywords and their effects are as follows:

BELL=NONE

Specifies which new messages should be accompanied by a warning bell character. By default, only broadcast messages are accompanied by a warning bell.

KEYPAD=EDT

Specifies whether the key definitions should be similar to EDT or EVE. Note that EVE key definitions do not use the numeric keypad on VT200 (or higher) terminals;numeric keypads on VT200-series (or higher) terminals emulate EDT key definitions, regardless of the keypad mode you choose.

MENU=[NO]GRAPHICS

Lets you choose between graphic characters and nongraphic characters in the display of a menu. The initial setting is MENU=GRAPHICS. If the terminal characteristics do not include DEC_CRT, LSE uses nongraphic characters, regardless of the setting of this mode.

Graphic characters currently require more screen repainting than do nongraphic characters, so you might want to use SET MODE MENU=NOGRAPHICS if you are working at a low baud rate.

PENDING_DELETE

Specifies whether a selection in a user buffer should be deleted when the user inserts text. The initial setting is NOPENDING_DELETE. PENDING_DELETE is disabled for a selection made with SELECT ALL. You can use the UNERASE SELECTION command to restore deleted text.

TAB=VISIBLE

Specifies whether tabs should appear as blanks, or a combination of the HT (horizontal tab) symbol and dots (“HT.....”).

TABS=[NO]HARD

Specifies whether tab or space characters are used for tabulation. HARD (the default) specifies tab characters, whereas NOHARD specifies space characters.

UNDO=ON

Specifies whether UNDO processing is enabled (ON is the default).

Description

The SET MODE command establishes the status of warning bells, keypad emulation, selected range, UNDO processing, tab appearance, tab characters, and the use of graphic characters.

DECwindows Interface Equivalent

Pull-down menu: Options → Global Attributes . . .

Related Commands

SHOW MODE

Examples

1. LSE> ***SET MODE BELL=NOBROADCAST***

   Prevents the warning bell from sounding when broadcast messages appear in the LSE message buffer.

2. LSE> ***SET MODE KEYPAD=EVE***

   Sets key definitions to be the same as those used with EVE.

3. LSE> ***SET MODE PENDING_DELETE***

   Causes a selection to be deleted when the user inserts text into a user buffer.

4. LSE> ***SET MODE TAB=INVISIBLE***

   Causes tabs to be displayed as blanks.

5. LSE> ***SET MODE TABS=NOHARD***

Causes tabs to be implemented using space characters.

6. LSE> ***SET MODE UNDO=OFF***

   Switches off UNDO processing.

SET MODIFY

SET MODIFY — Sets the buffer status to modifiable.

Format

SET MODIFY

Qualifier	Defaults
/BUFFER=buffer	

Qualifier

/BUFFER=buffer

Indicates the buffer to be set as modifiable. The current buffer is the default.

Description

The SET MODIFY command changes the status of the current buffer, or the buffer specified, from unmodifiable to modifiable.

Users of the DECwindows interface can cycle through Insert, Overstrike, and Unmodifiable by selecting the status line button and pressing MB1. If the status line shows Insert and Overstrike, then the buffer is modifiable.

DECwindows Interface Equivalent

Buffer status line $\left\{\begin{matrix}\text{Insert}\\\text{Overstrike}\\\text{Unmodifiable}\end{matrix}\right\}$

Pull-down menu: Options > Buffer Attributes . . .

Related Commands

SET CMS

SET DIRECTORY

SET NOMODIFY

Example

LSE> ***SET MODIFY***

Enables you to modify a file that you previously brought into the current buffer as Read-only.

SET NOAUTO_ERASE

SET NOAUTO_ERASE — Disables automatic erasing of placeholders in the specified buffer.

Format

SET NOAUTO_ERASE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer for which automatic erasing is to be disabled. The default is the current buffer.

Description

The SET NOAUTO_ERASE command prevents LSE from automatically erasing the placeholder the cursor is on when you type a character over that placeholder in the specified buffer.

Initially, LSE is set to automatically erase placeholders.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

SET AUTO_ERASE

SHOW BUFFER [DECwindows interface]

Example

```
LSE> SET NOAUTOERASE/BUFFER=USER.TXT
```

Disables automatic erasing of placeholders in the buffer USER.TXT.

SET NOJOURNALING

SET NOJOURNALING — Disables buffer journaling for the specified buffers.

Format

SET NOJOURNALING [buffer-name]

Qualifier	Defaults
/ALL	

Qualifier

/ALL

Specifies that all of the LSE buffer-change journal files should be closed and buffer-change journaling halted for those buffers. If you specify the /ALL qualifier, you cannot specify the *buffer-name* parameter.

Parameter

buffer-name

Specifies the name of the buffer that no longer has an associated buffer-change journal file. If you omit this parameter,the default is the current buffer.

Description

The SET NOJOURNALING command terminates buffer-change journaling for the specified buffer. Any subsequent changes to the buffer are not journaled, unless you use the SET JOURNALING command to enable buffer-change journaling.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

RECOVER BUFFER

SET JOURNALING

Example

```
LSE> SET NOJOURNALING login.com
```

Terminates buffer-change journaling for the buffer *login.com*.

SET NOLANGUAGE

SET NOLANGUAGE — Disassociates the language associated with the specified buffer.

Format

SET NOLANGUAGE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose associated language you want to disassociate. The current buffer is the default.

Description

The SET NOLANGUAGE command disassociates the language currently in effect from the specified buffer.

System buffers, such as $REVIEW, $MESSAGES, or $HELP, have no languages associated with them; if you attempt to use this command with system buffers,you receive an error message.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

RECOVER BUFFER

SET JOURNALING

Example

```
LSE> SET NOLANGUAGE
```

Disassociates the currently associated language from the current buffer.

SET NOLIBRARY

SET NOLIBRARY — Removes the specified SCA libraries from the current list of active libraries.

Format

SET NOLIBRARY [library-spec[, . . .]]

Qualifier	Default
/[NO]LOG	/LOG

Qualifier

/LOG (D)
/NOLOG

Indicates whether LSE reports removal of the libraries from the active list.

Parameter

library-spec[, . . .]

Specifies the libraries to be removed from the current active libraries list. If you omit this parameter, SCA removes all the active libraries from the list.

Description

The SET NOLIBRARY command enables you to selectively discard or purge specific SCA libraries from an active library list.

Related Commands

SET LIBRARY

Example

```
LSE> SET NOLIBRARY PROJ:[USER.LIB1],PROJ:[USER.LIB2]
```

Removes the specified libraries from the current active libraries list.

See the chapter about using SCA libraries in the *VSI DECset for OpenVMS Guide to Source Code Analyzer* for additional examples.

SET NOMODIFY

SET NOMODIFY — Sets a buffer to Read-only (unmodifiable).

Format

SET NOMODIFY

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer that is to be set to Read-only. The default is the current buffer.

Description

The SET NOMODIFY command sets a buffer to read-only (unmodifiable). After entering this command, you cannot change the buffer's contents until you enter a SET MODIFY command.

Users of the DECwindows interface can cycle through Insert, Overstrike, and Unmodifiable by selecting the status line button and pressing MB1.

DECwindows Interface Equivalent

Buffer status line {Insert / Overstrike / Unmodifiable}

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

SET MODIFY

SHOW BUFFER [DECwindows interface]

Example

```
LSE> SET NOMODIFY
```

Prevents you from modifying text that you had previously brought into the current buffer as modifiable.

SET NOOUTPUT_FILE

SET NOOUTPUT_FILE — Disassociates the buffer from any output file.

Format

SET NOOUTPUT_FILE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose output file is to be changed. The default is the current buffer.

Description

The SET NOOUTPUT_FILE command disassociates the specified buffer from any output file. LSE uses output file associations when writing the buffer out to a file; thus, when you enter the SET NOOUTPUT_FILE command and then enter a COMPILE, EXIT, or WRITE command, you must supply LSE with a file name.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

COMPILE

EXIT

SET OUTPUT_FILE

SHOW BUFFER [DECwindows interface]

WRITE

Example

```
LSE> SET NOOUTPUT_FILE
```

Disassociates the current buffer from any output file. You must specify a file name to write the buffer to if you subsequently enter an EXIT or WRITE command.

SET NOOVERVIEW

SET NOOVERVIEW — Disables overview operations in the specified buffer.

Format

SET NOOVERVIEW

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer in which overview operations are to be disabled. The default is the current buffer.

Description

The SET NOOVERVIEW command disables the use of overview operations in the specified buffer. This disables the COLLAPSE, FOCUS, and VIEW SOURCE commands, and the use of the EXPAND command on an overview line.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

COLLAPSE

EXPAND

FOCUS

SET OVERVIEW

VIEW SOURCE

SET NOSOURCE_DIRECTORY

SET NOSOURCE_DIRECTORY — Specifies a directory or directories to be removed from the list of source directories.

Format

SET NOSOURCE_DIRECTORY [directory-spec [,directory-spec] . . .]

Parameter

directory-spec [,directory-spec] . . .

Specifies a list of directory specifications to be removed from the list of source directories. If you do not specify any parameter, LSE removes all directories from the list of source directories.

Description

The SET NOSOURCE_DIRECTORY command removes the directories you specify from the list of source directories. If you do not specify any directories, LSE removes all directories in the source list from that list.

Related Command

SET SOURCE_DIRECTORY

Examples

1. LSE> ***SET NOSOURCE [PROJECT_DIRECTORY]***

   Removes the directory PROJECT_DIRECTORY from the list of source directories.

2. LSE> ***SET NOSOURCE/READ_ONLY [LIBRARY_DIRECTORY]***

   Removes the directory LIBRARY_DIRECTORY from the set of read-only directories.

SET NOWRAP

SET NOWRAP — Disables wrapping of the current line in the specified buffer.

Format

SET NOWRAP

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer for which wrapping is to be disabled. The default is the current buffer.

Description

The SET NOWRAP command prevents the ENTER SPACE command (bound to the space bar by default) from performing a wrap operation on the current line in the specified buffer.

Initially, wrapping of the current line is disabled.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

ENTER LINE

ENTER SPACE

SET WRAP

SHOW BUFFER [DECwindows interface]

SET OUTPUT_FILE

SET OUTPUT_FILE — Establishes the output file associated with the buffer.

Format

SET OUTPUT_FILE file-spec

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose output file is to be changed. The default is the current buffer.

Parameter

file-spec

Indicates the file specification for the output file.

Description

The SET OUTPUT_FILE command associates the specified output file with the specified buffer. LSE uses output file associations when writing the buffer out to a file; this happens when you enter a COMPILE, EXIT, or WRITE command.

This command does not cause the buffer to be written to a file. You might also need to use the SET WRITE command.

DECwindows Interface Equivalents

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

ENTER LINE

ENTER SPACE

SET WRAP

SHOW BUFFER [DECwindows interface]

Example

```
LSE> SET OUTPUT_FILE USER.TXT
```

Associates the output file USER.TXT with the current buffer. When you enter an EXIT or WRITE command, LSE writes the contents of that buffer to the file USER.TXT without prompting you for a file name.

SET OVERSTRIKE

SET OVERSTRIKE — Sets the text-entry mode of the specified buffer to overstrike mode.

Format

SET OVERSTRIKE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose text-entry mode is to be changed. The default is the current buffer.

Description

The SET OVERSTRIKE command sets the mode of the specified buffer to overstrike mode. When you set this mode, typing a character replaces that character at the current cursor position. Pressing the Delete key replaces the character to the left of the cursor with a blank space.

The status line of each window displays the current text-entry mode for the associated buffer.

Users of the DECwindows interface can cycle through Insert, Overstrike, and Unmodifiable by selecting the status line button and pressing MB1.

DECwindows Interface Equivalent

Buffer status line {Insert / Overstrike / Unmodifiable}

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

CHANGE TEXT_ENTRY_MODE

SET INSERT

SET OVERVIEW

SET OVERVIEW — Enables overview operations in the specified buffer.

Format

SET OVERVIEW

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer in which overview operations are to be enabled. The default is the current buffer.

Description

The SET OVERVIEW command enables the use of overview operations in the specified buffer. This enables the COLLAPSE,FOCUS and VIEW SOURCE commands, as well as the use of the EXPAND command on an overview line.

By default, overview operations are allowed in a buffer when it is created. LSE disables overview operations in some system buffers that it creates. To see the current setting, use the SHOW BUFFER/FULL command.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

COLLAPSE

EXPAND

FOCUS

SET NOOVERVIEW

VIEW SOURCE

SET READ_ONLY

SET READ_ONLY — Instructs LSE not to write the specified buffer to a file when you exit from LSE, or when you enter a COMPILE command.

Format

SET READ_ONLY

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose read-only or write state is to be changed. The default is the current buffer.

Description

The SET READ_ONLY command prevents LSE from writing the contents of the specified buffer to a file when you exit from LSE or enter a COMPILE command. The LSE status line displays the read-only or write state.

Users of the DECwindows interface can switch between Write and Read-only by selecting the status line button and pressing MB1.

DECwindows Interface Equivalent

Buffer status line $\left\{\begin{matrix}\text{Write} \\ \text{Read-only}\end{matrix}\right\}$

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

SET MODIFY

SET NOMODIFY

SET WRITE

SHOW BUFFER [DECwindows interface]

SET REVERSE

SET REVERSE — Sets the current direction of the specified buffer to reverse.

Format

SET REVERSE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose direction is to be set to reverse. The default is the current buffer.

Description

The SET REVERSE command sets the current direction of the specified buffer to reverse. The status line displays the current direction.

Users of the DECwindows interface can switch between Forward and Reverse by selecting the status line button and pressing MB1.

Keypad Equivalent

Key	Keypad Equivalent
KP5 **REVERSE**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

DECwindows Interface Equivalent

Buffer status line $\left\{\begin{matrix}\text{Forward}\\\text{Reverse}\end{matrix}\right\}$

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

CHANGE DIRECTION

SET FORWARD

SET RIGHT_MARGIN

SET RIGHT_MARGIN — Sets the right margin for the specified buffer.

Format

SET RIGHT_MARGIN column-number

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose right margin is to be changed. The default is the current buffer.

Parameter

column-number

Specifies the column for the right margin. The value must be an integer greater than the value set for the left margin.

Description

The SET RIGHT_MARGIN command sets the right margin of the specified buffer to the column number you specify. By default, the right margin is set at column 80.

The right margin controls where LSE wraps words when you type new text. The FILL and ENTER SPACE commands also use this setting. To find out the setting of the right margin, use the SHOW BUFFER command.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes

Related Commands

SET LEFT_MARGIN

SHOW BUFFER [DECwindows interface]

Example

```
LSE> SET RIGHT_MARGIN 65
```

Sets the right margin in the current buffer at column 65.

SET SCREEN

SET SCREEN — Sets specified characteristics of the screen.

Format

SET SCREEN keyword-list

Parameter

keyword-list

Indicates the screen characteristics to be set. The screen keywords are as follows:

BALANCE_WINDOWS (D)

Specifies how LSE manages window length. If you specify BALANCE_WINDOWS, LSE adjusts all the window lengths on the screen to be, as nearly as possible, of equal lengths. This is the default value. If you specify NOBALANCE_WINDOWS, LSE splits the current window in half when it needs a new window, which leaves all the other window lengths unchanged.

HEIGHT=n

Specifies the number of lines on the screen. The height, *n*, must be an integer in the range 11 through 62.

MAXIMUM_WINDOW_NUMBER=n

Specifies the maximum number of windows LSE creates when it displays information in a window as a result of entering one of the following commands:

FIND
GOTO DECLARATION
GOTO SOURCE
INSPECT
REVIEW

LSE uses the MAXIMUM_WINDOW_NUMBER and MINIMUM_WINDOW_LENGTH settings to determine whether to add a window to the screen,or reuse an existing window. LSE checks both settings and creates a new window only if both conditions are met.

The default value for MAXIMUM_WINDOW_NUMBER is 3. Specifying a value of 2 produces the two-window behavior previously associated with the commands listed under this keyword.

MINIMUM_WINDOW_LENGTH=n

Specifies a lower bound on the windows LSE creates. When you need to map a buffer to a window, LSE creates a new window as long as the window is not shorter than *n*.

LSE uses the MINIMUM_WINDOW_LENGTH and MAXIMUM_WINDOW_NUMBER settings to determine whether to add a window to the screen)or reuse an existing window. LSE checks both settings and creates a new window only if both conditions are met.

WIDTH=n

Specifies the number of characters on each input or output line. The width, *n*, must be an integer in the range 1 through 252.

If you specify a width greater than 80, LSE sets the terminal to132-character mode. The initial setting is 80 characters.

WINDOW=n

Specifies the number of windows to display on the screen.

If you change the number of windows from one to two, LSE displays the current buffer in both windows. If you change the number of windows from two to one, LSE displays the current buffer in the single window. The initial setting is one window.

DECwindows Interface Equivalent

Pull-down menu: Options → Window Attributes

Related Commands

SHOW SCREEN

Examples

1. LSE> ***SET SCREEN WIDTH=132***

   Sets your terminal to 132-character mode.

2. LSE> ***SET SCREEN MINIMUM_WINDOW_LENGTH=5,BALANCE_WINDOWS***

   For automatic window creation on a 24-line terminal, the keyword MINIMUM_WINDOW_LENGTH=5 allows up to four windows and the keyword BALANCE_WINDOWS causes the editor to keep all the windows approximately equal in length.

SET SCROLL_MARGINS

SET SCROLL_MARGINS — Delimits the lines at which the cursor triggers scrolling.

Format

SET SCROLL_MARGINS top-line-count[%] bottom-line-count[%]

Parameters

top-line-count

Specifies the number of lines down from the top of a window at which you want downward scrolling to begin.

bottom-line-count

Specifies the number of lines up from the bottom of a window at which you want upward scrolling to begin.

%

Optionally specifies scroll margins as percentages of the window height, which are rounded to the nearest whole-line count. This is useful when you have a workstation with screens of varying sizes.

Description

The SET SCROLL_MARGINS command specifies the lines at the top and bottom of the window at which scrolling is triggered by moving the cursor to these lines.

The scroll margins you set apply to all windows in the current editing session.

Examples

1. LSE> ***SET SCROLL_MARGINS 2 3***

   Sets the scroll margins at 2 lines from the top and 3 lines from the bottom of all windows in the current editing session.

2. LSE> ***SET SCROLL_MARGINS 10% 15%***

   Sets the scroll margins at 10% from the top and 15% from the bottom of all windows in the current editing session.

SET SEARCH

SET SEARCH — Sets text search options.

Format

SET SEARCH keyword-list

Parameter

keyword-list

Indicates the search mode settings. The keywords are as follows:

AUTO_REVERSE

NOAUTO_REVERSE

Specifies whether LSE searches in the current direction only, or searches in the opposite direction if the string is not found in the current direction. The initial setting is NOAUTO_REVERSE.

CASE_SENSITIVE

NOCASE_SENSITIVE

Specifies whether the SEARCH command matches case exactly or is insensitive to character case. The initial setting is NOCASE_SENSITIVE.

DIACRITICAL_SENSITIVE

NODIACRITICAL_SENSITIVE

Specifies whether the SEARCH command matches characters with diacritical markings exactly or is insensitive to diacritical markings. The initial setting is DIACRITICAL_SENSITIVE.

PATTERN=OPENVMS

PATTERN=ULTRIX

PATTERN=TPU

Specifies either UNIX-style regular expressions, OpenVMS-style patterns or TPU-style patterns for the SEARCH/PATTERN command. The initial setting is OpenVMS.

SPAN_SPACE

NOSPAN_SPACE

Determines whether LSE matches blanks in the search string exactly(NOSPAN_SPACE), or allows each blank to match sequences of one or more characters containing blanks and tabs and, at most, a single line break (SPAN_SPACE). The initial setting is NOSPAN_SPACE.

Description

The SET SEARCH command sets preconditions for matching text when you enter the SEARCH command.

DECwindows Interface Equivalent

Pull-down menu: Options → Search Attributes . . .

Related Commands

SEARCH

SHOW SEARCH

Example

```
LSE> SET SEARCH CASE_SENSITIVE
```

Directs LSE to match case exactly when you enter a SEARCH command.

SET SELECT_MARK

SET SELECT_MARK — Marks a position as one end of a selected range.

Format

SET SELECT_MARK

Description

The SET SELECT_MARK command marks a position as one end of a selected range. The selected range is the text between the select marker and the current cursor position; it is denoted by a reverse video display. This command is not valid if the select marker has already been set.

Keypad Equivalent

Key	Keypad Equivalent
Keypad period (.) **SELECT**	EDT LK201, EDT VT100, EVE LK201

Key	Keypad Equivalent
None	EVE VT100

DECwindows Interface Equivalent

Pull-down menu: Navigate → Mark . . .

Related Commands

CANCEL SELECT_MARK

TOGGLE SELECT_MARK

SET SOURCE_DIRECTORY

SET SOURCE_DIRECTORY — Specifies a search list of directories to be used to find source files.

Format

SET SOURCE_DIRECTORY directory-spec [,directory-spec] . . .

Qualifiers	Defaults
/AFTER[=directory-spec]	/AFTER
/BEFORE[=directory-spec]	/AFTER

Qualifiers

/AFTER (D)
/AFTER[=directory-spec]

Specifies that LSE should insert the directory or directories specified into the list of source directories in back of the directory you specify as the value on the qualifier. If you do not specify a *directory-spec* value, LSE adds the directory or directories to the end of the list.

If you do not specify either the /AFTER qualifier or the /BEFORE qualifier, LSE replaces the entire directory list.

/BEFORE
/BEFORE[=directory-spec]

Specifies that LSE should insert the directory or directories specified into the list of source directories in front of the directory you specify as the value on the qualifier. If you do not specify a *directory-spec*value, the directory or directories are added at the front of the list.

If you do not specify either the /BEFORE qualifier or the /AFTER qualifier, LSE replaces the entire directory list.

Parameter

directory-spec [,directory-spec] . . .

Specifies one or more directory specifications. You can specify CMS$LIB as one directory specification; however, you might not get the results you expect if you set CMS$LIB as a source directory and do not enter the CMS command SET LIBRARY.

Description

The SET SOURCE_DIRECTORY command specifies the directories LSE uses to find source files when you enter the commands GOTO FILE, GOTO SOURCE, and READ.

The GOTO FILE and READ commands use this list of directories if you do not specify a directory for the file specified on the GOTO FILE or READ command.

The GOTO SOURCE command uses this list of directories if LSE does not find the source file specified in the SCA data file or the diagnostics file.

The logical name LSE$SOURCE stores the list of source directories.

Related Commands

SET CMS

SET NOSOURCE_DIRECTORY

Example

```
LSE> SET SOURCE_DIRECTORY [],[MY_SOURCE_DIRECTORY],-
_LSE> [PROJECT_SOURCE_DIRECTORY],CMS$LIB
```

Directs LSE to search for sources first in the current directory, then in the user's source directory, then in the project source directory, and finally in CMS$LIB.

SET TAB_INCREMENT

SET TAB_INCREMENT — Specifies logical tab stops in the specified buffer.

Format

SET TAB_INCREMENT number

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose tab increment is to be changed. The default is the current buffer.

Parameter

number

Specifies the interval for setting tab stops.

Description

The SET TAB_INCREMENT command specifies the number of columns between the tab stops for the specified buffer. Tab stops are set beginning with column 1. All previous tab stops are cleared.

Related Commands

ENTER TAB

SHOW BUFFER [DECwindows interface]

TAB

Example

```
LSE> SET TAB_INCREMENT 4
```

Sets tab stops in columns 1, 5, 9, 13, and so on.

SET WRAP

SET WRAP — Enables wrapping in the specified buffer. LSE automatically splits the current line at the right-margin setting when you type text past the right margin.

Format

SET WRAP

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer for which wrapping is to be enabled. The default is the current buffer.

Description

The SET WRAP command enables the ENTER SPACE and ENTER LINE commands to perform a wrap operation in the specified buffer.

Initially, wrapping is disabled.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

ENTER LINE

ENTER SPACE

SET NOWRAP

SHOW BUFFER [DECwindows interface]

SET WRITE

SET WRITE — Instructs LSE to write the contents of the specified buffer to a file when you exit from LSE or enter a COMPILE command.

Format

SET WRITE

Qualifier	Defaults
/BUFFER=buffer-name	

Qualifier

/BUFFER=buffer-name

Indicates the buffer whose read-only or write state is to be changed. The default is the current buffer.

Description

The SET WRITE command reverses the action of the SET READ_ONLY command. When you exit from LSE or enter a COMPILE command, LSE writes the contents of the specified buffer to a file. The status line displays the setting of the read-only or write state.

If the specified buffer is unmodifiable, entering a SET WRITE command is also equivalent to entering a SET MODIFY command. If the directory for the file associated with the buffer is read-only, LSE displays a message informing you of that fact.

Users of the DECwindows interface can switch between Write and Read-only by selecting the status line button and pressing MB1.

DECwindows Interface Equivalent

Buffer status line $\begin{Bmatrix} \text{Forward} \\ \text{Reverse} \end{Bmatrix}$

Pull-down menu: Options > Buffer Attributes . . .

Related Commands

SET OUTPUT_FILE

SET READ_ONLY

SHOW BUFFER [DECwindows interface]

SHIFT

SHIFT — Shifts the window horizontally to the left or right one column.

Format

SHIFT

Qualifiers	Defaults
/CURRENT	/CURRENT
/FORWARD	/CURRENT
/REVERSE	/CURRENT

Qualifiers

/CURRENT (D)

Specifies the current direction for the shift.

/FORWARD

Shifts the window to the right so you can view formerly hidden text to the right of the original text.

/REVERSE

Shifts the window to the left so you can view any text hidden by a SHIFT/FORWARD command.

Description

The SHIFT command shifts or moves the display window horizontally to the left or right one column. The SHIFT qualifiers refer to the direction of window movement with respect to the text. When used with a repeat count, the value of the repeat count determines the extent of the shift (see the REPEAT command).

Users of the DECwindows interface can achieve similar results by using the horizontal scroll bar.

Example

```
LSE> REPEAT 3 SHIFT/FORWARD
```

Moves the display window 3 columns to the right.

SHOW ADJUSTMENT

SHOW ADJUSTMENT — Displays the characteristics of specified adjustments.

Format

SHOW ADJUSTMENT [adjustment-name]

Qualifier	Defaults
/LANGUAGE=language-name	

Qualifier

/LANGUAGE=language-name

Associates a language with the specified adjustments. If you do not specify a language, LSE displays information about adjustments associated with the correct language. If you specify /LANGUAGE=*, LSE displays information on any adjustment that matches the adjustment name, regardless of the language for which it is defined.

Parameter

adjustment-name

Specifies which adjustments are to be shown. If you omit this parameter, LSE assumes you have specified a wildcard adjustment name.

Description

The SHOW ADJUSTMENT command displays the definitions and characteristics of adjustments.

Related Commands

DEFINE ADJUSTMENT

DELETE ADJUSTMENT

EXTRACT ADJUSTMENT

Example

```
LSE> SHOW ADJUSTMENT then
```

Displays all the characteristics defined for the adjustment *then*.

SHOW ALIAS

SHOW ALIAS — Displays information on the specified alias.

Format

SHOW ALIAS [alias-name]

Qualifiers	Default
/BRIEF	See text
/FULL	See text
/LANGUAGE=language-name	

Qualifiers

/BRIEF

Causes LSE to display (in tabular format) the alias name and equivalent string.

If you specify a wildcard expression for the parameter or if LSE assumes one, /BRIEF is the default.

/FULL

Causes LSE to display the alias name and equivalent string in list format.

If you specify an explicit name for the parameter, /FULL is the default.

/LANGUAGE=language-name

Specifies the language associated with the alias. The default is the current language.

Parameter

alias-name

Specifies the name of the alias whose characteristics are to be displayed. If this parameter is omitted, a wildcard alias name is assumed.

Description

The SHOW ALIAS command displays information on an alias you defined using the DEFINE ALIAS command.

Related Commands

DEFINE ALIAS

Example

```
LSE> SHOW ALIAS
```

Displays one line of information for each of the aliases you have currently defined.

SHOW BUFFER

SHOW BUFFER — Displays the characteristics of one or more buffers.

Format

SHOW BUFFER [buffer-name]

Qualifiers	Defaults
/ALL_BUFFERS	/USER_BUFFERS
/BRIEF	See text
/FULL	See text

Qualifiers	Defaults
/SYSTEM_BUFFERS	/USER_BUFFERS
/USER_BUFFERS	/USER_BUFFERS

Qualifiers

/ALL_BUFFERS

Specifies all buffers to be displayed when a wildcard buffer name is specified or assumed. LSE ignores this qualifier if you specify an explicit buffer name.

/BRIEF

Causes the current window to display (in tabular format) the name, number of text lines, and information about whether the buffer is modified, compiled,reviewed, or modifiable.

If you move the cursor to a line containing a buffer name and press the Select key, LSE performs a GOTOBUFFER command for that buffer. If you move the cursor to a line containing a buffer name and press the Remove key, LSE performs a DELETE BUFFER command for that buffer. In DECwindows mode, you can perform a GOTO BUFFER for a buffer displayed in the list by pressing Return on the line containing the buffer name.

If you specify a wildcard expression, or if LSE assumes one, /BRIEF is the default.

/FULL

Causes LSE to list all the information available about each specified buffer, including associated input and output files, language, and all the buffer attributes that you can set, such as margins and text-entry mode.

If you specify an explicit buffer, /FULL is the default.

/SYSTEM_BUFFERS

Specifies that only system buffers be displayed when a wildcard buffer name is specified or assumed. LSE ignores this qualifier if you specify an explicit buffer name.

/USER_BUFFERS (D)

Specifies that only user buffers be displayed when a wildcard buffer name is specified or assumed. LSE ignores this qualifier if you specify an explicit buffer name.

Parameter

buffer-name

Specifies the name of the buffers whose characteristics are to be displayed. If you specify a null buffer name (" "), the current buffer is assumed. If this parameter is omitted, a wildcard buffer name is assumed.

Description

The SHOW BUFFER command displays information about the specified buffers.

DECwindows Interface Equivalent

SHOW BUFFER

Pull-down menu: Show → Show Buffer *

SHOW BUFFER /FULL ""

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

NEXT BUFFER

PREVIOUS BUFFER

SET AUTO_ERASE

SET LEFT_MARGIN

SET NOAUTO_ERASE

SET NOOUTPUT_FILE

SET NOWRAP

SET OUTPUT_FILE

SET READ_ONLY

SET RIGHT_MARGIN

Example

```
LSE> SHOW BUFFER/SYSTEM
```

Displays the name, number of text lines, and status (read-only or modifiable) for each system buffer.

SHOW CMS

SHOW CMS — Displays the current CMS settings, which are the initial settings unless you have changed them using the SET CMS command.

Format

SHOW CMS

Description

The SHOW CMS command lists all the CMS settings specified by the qualifiers to the SET CMS command. If you have not entered a SET CMS command, the listed CMS settings reflect initial conditions.

For users of the DECwindows interface, the SHOW CMS command displays a CMS Attribute dialog box to let you change the current CMS settings.

DECwindows Interface Equivalent

Pull-down menu: Options → CMS . . .

Related Commands

SET CMS

SHOW COMMAND

SHOW COMMAND — Displays the characteristics of the specified user-defined command.

Format

SHOW COMMAND [command-name]

Parameter

command-name

Specifies the name of the command whose characteristics are to be displayed. If you omit this parameter, LSE displays information on all user-defined commands.

Description

The SHOW COMMAND command displays the characteristics of a command you have defined using the DEFINE COMMAND command.

DECwindows Interface Equivalent

Pull-down menu: Show → Show Command *

Related Commands

DEFINE COMMAND

Example

```
LSE> SHOW COMMAND
```

Displays the command definition for each user-defined command.

SHOW DEFAULT_DIRECTORY

SHOW DEFAULT_DIRECTORY — Displays the current default device and directory.

Format

SHOW DEFAULT_DIRECTORY

Description

The SHOW DEFAULT_DIRECTORY command displays the current device and directory names, along with any equivalence strings. You can change the default with the LSE command SET DEFAULT_DIRECTORY.

Related Commands

SET DEFAULT_DIRECTORY

Example

```
LSE> SHOW DEFAULT_DIRECTORY
```

Displays the current device and directory names.

SHOW DIRECTORY

SHOW DIRECTORY — Displays the setting of the SET DIRECTORY command.

Format

SHOW DIRECTORY

Description

The SHOW DIRECTORY command displays the list of directories specified by the SET DIRECTORY command.

Related Commands

SET DIRECTORY

SHOW KEY

SHOW KEY — Displays the definitions bound to the normal and GOLD states of any defined key.

Format

SHOW KEY key-specifier

Qualifiers	Defaults
/BRIEF	/BRIEF

Qualifiers	Defaults
/FULL	/BRIEF

Qualifiers

/BRIEF (D)

Indicates how much information you want displayed. The /BRIEF qualifier instructs LSE to display only key names and the commands associated with them.

/FULL

Indicates how much information you want displayed. The /FULL qualifier instructs LSE to display topics, legends, and remarks, as well as the key names and commands.

Parameter

key-specifier

Specifies the name of the key whose definitions are to be displayed. You can use a wildcard character on the command line to specify all defined keys or a group of related keys. If you press the Return key before specifying a key, LSE supplies quotation marks to any specifier you type at the prompt. Therefore, LSE interprets an asterisk specified at the prompt as the asterisk key on the keyboard, not as a wildcard character.

To specify key combinations beginning with the PF1 key, use the prefix GOLD/. To specify combinations by using the control key, use the form Ctrl/ *x*,where *x* can be the letters A through Z.

Description

The SHOW KEY command displays the definitions bound to the normal and GOLD states of any or all keyboard keys. This includes both the default bindings and those keys you have bound using the DEFINE KEY command.

The SHOW KEY command accepts key names that are valid for the DEFINE KEY command if you have used the following syntax for the key being defined:

```
LSE> DEFINE KEY "CTRL/A" "SHOW BUFFER"
```

DECwindows Interface Equivalent

Pull-down menu: Show → Show Key *

Related Commands

DEFINE KEY

DELETE KEY

Examples

1. `LSE> SHOW KEY PF2`

Displays the definitions currently bound to the PF2 key.

2. `LSE> SHOW KEY CTRL*`

Displays the definitions currently bound to all key sequences that begin with Ctrl.

SHOW KEYWORDS

SHOW KEYWORDS — Displays the characteristics of the specified keyword list.

Format

SHOW KEYWORDS [keyword-list-name]

Qualifiers	Defaults
/BRIEF	/BRIEF
/FULL	/BRIEF

Qualifiers

/BRIEF (D)

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name of the specified keyword list.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes LSE to display all the information available about the specified keyword list, as specified by the current DEFINEKEYWORDS command (see the list of qualifiers for the DEFINE KEYWORDS command).

Parameter

keyword-list-name

Specifies the keyword lists about which information is wanted. By default, LSE displays information about the keyword list associated with the current buffer.

Description

The SHOW KEYWORDS command displays the characteristics of a specified keyword list. The keyword list must be known to LSE.

Related Commands

DEFINE KEYWORDS

DELETE KEYWORDS

EXTRACT KEYWORDS

Example

```
LSE> SHOW KEYWORDS author_name
```

Displays the characteristics associated with *author_name*.

SHOW LANGUAGE

SHOW LANGUAGE — Displays the characteristics of the specified language.

Format

SHOW LANGUAGE [language-name]

Qualifiers	Defaults
/BRIEF	/FULL
/FULL	/FULL

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and file type of the specified language.

/FULL (D)

Indicates how much information you want displayed. The /FULL qualifier causes LSE to display all the information available about the specified language, as specified by the current DEFINELANGUAGE command (see the list of qualifiers for the DEFINE LANGUAGE command).

Parameter

language-name

Specifies the languages about which information is wanted. By default, LSE displays information about the language associated with the current buffer.

Description

The SHOW LANGUAGE command displays the characteristics of a specified language. The language must be known to LSE.

Related Commands

DEFINE LANGUAGE

MODIFY LANGUAGE

SET LANGUAGE

Example

```
LSE> SHOW LANGUAGE Pascal
```

Displays the compiler, file type, punctuation, and other characteristics associated with the programming language Pascal.

SHOW LIBRARY

SHOW LIBRARY — Displays the directory specification for all active SCA libraries.

Format

SHOW LIBRARY

Qualifiers	Defaults
/BRIEF	/BRIEF
/FULL	/BRIEF

Qualifiers

/BRIEF (D)

Displays only the directory specification for all active libraries.

/FULL

Displays all information about all active SCA libraries.

Description

The SHOW LIBRARY command displays the directory specifications for all active SCA libraries.

Related Commands

CREATE LIBRARY

SET LIBRARY

SET NOLIBRARY

Example

```
$ SCA SHOW LIBRARY
```

Displays the location of the current library.

SHOW MARK

SHOW MARK — Displays the setting of the specified mark.

Format

SHOW MARK [marker-name]

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and associated buffer for each marker currently set.

If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes LSE to list all the information available about each specified marker, including the associated text.

If you specify an explicit marker for the parameter, /FULL is the default.

Parameter

marker-name

Specifies the name of the marker whose characteristics are to be displayed. If you omit this parameter, LSE displays the names of all the markers you have set.

Description

The SHOW MARK command displays the names of markers associated with the current buffer.

DECwindows Interface Equivalent

Pull-down menu: Show → Show Mark *

Related Command

SET MARK

Example

```
LSE> SHOW MARK
```

Lists the currently set marker names and their associated buffers.

SHOW MAX_UNDO

SHOW MAX_UNDO — Shows the maximum number of UNDO operations you can perform for a specific buffer.

Format

SHOW MAX_UNDO

Qualifiers	Defaults
/BUFFER=buffer-name	

Qualifiers

/BUFFER=buffer-name

Indicates the buffer whose maximum undo number is to be displayed. The default is the current buffer.

Description

Shows the maximum number of undo operations that you can undo for a specific buffer. If you have not set a maximum number with the SET MAX_UNDO command, the SHOW MAX_UNDO command displays the default value of 100.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

REDO

SET MAX_UNDO

SET MODE UNDO=OFF

SET MODE UNDO=ON

UNDO

Example

```
LSE> SHOW MAX_UNDO
```

Displays the maximum number of UNDO operations that you can perform on the current buffer.

SHOW MODE

SHOW MODE — Displays the current settings for modes set with the SET MODE command.

Format

SHOW MODE

Description

The SHOW MODE command displays the current mode settings for keywords used with the SET MODE command.

For users of the DECwindows interface, the SHOW MODE command uses the Global Attribute dialog box to display the modes. This dialog box permits you to change the mode settings.

DECwindows Interface Equivalent

Pull-down menu: Options → Buffer Attributes . . .

Related Commands

SET MODE

Example

```
LSE> SHOW MODE
```

Displays the current editing-mode status for warning bells, keypad mode, select range, UNDO processing, tab appearance, tab characters, and menu-display characters.

SHOW MODULE

SHOW MODULE — Displays information about SCA library modules.

Format

SHOW MODULE [module-name[, . . .]]

Qualifiers	Defaults
/ALL	/VISIBLE
/BRIEF	See text
/FULL	See text
/HIDDEN	/VISIBLE
/LIBRARY=library-spec	/LIBRARY=*
/OUTPUT[=file-spec]	
/VISIBLE	/VISIBLE

Qualifiers

/ALL

Specifies that SCA display both hidden and visible modules.

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes SCA to display selected information about each specified module in tabular format. For an example, see the chapter about getting started with SCA in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

If you specify a wildcard expression for the parameter, or if SCA assumes one, /BRIEF is the default.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes SCA to list all information available about each specified module. For an example, see the chapter about getting started with SCA in the *VSI DECset for OpenVMS Guide to Source Code Analyzer*.

If you specify an explicit name for the parameter, /FULL is the default.

/HIDDEN

Specifies that SCA display only hidden modules.

/LIBRARY=library-spec
/LIBRARY=* (D)

Specifies an SCA library containing the module to be displayed. The library must be one of the current SCA libraries established by a SET LIBRARY command.

If you do not specify the /LIBRARY qualifier, SCA assumes you have specified all current SCA libraries.

/OUTPUT[=file-spec]

Directs command output to a file rather than to the $SHOW buffer. The default output file specification is SCA.LIS.

/VISIBLE (D)

Specifies that SCA display only visible modules.

Parameter

[module-name[, . . .]]

Specifies the modules to be displayed. If you omit this parameter, SCA displays all modules. You can specify wildcard module names.

Description

The SHOW MODULE command displays information about modules in SCA libraries

Related Commands

SET LIBRARY

Example

```
$ SCA SHOW MODULE
```

Displays all of the source module information from the library in an abbreviated form. (/BRIEF is the default.)

SHOW PACKAGE

SHOW PACKAGE — Displays the characteristics of the specified packages.

Format

SHOW PACKAGE package-name

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name of each specified package.

If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes LSE to display all the information available about each specified package, as specified by the current DEFINE PACKAGE command (seethe list of qualifiers for the DEFINE PACKAGE command).

If you specify an explicit name for the parameter, /FULL is the default.

Parameter

package-name

Specifies the name of the package to be displayed. You can use wildcards. If you omit this parameter, LSE displays the status of all known packages.

Description

The SHOW PACKAGE command displays the status of the specified packages. By default, LSE gives a brief description.

Related Commands

DEFINE PACKAGE

Example

```
LSE> SHOW PACKAGE system_services
```

Displays all the characteristics defined for the package system_services.

SHOW PARAMETER

SHOW PARAMETER — Displays the characteristics of the specified parameters.

Format

SHOW PARAMETER [parameter-name]

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text
/LANGUAGE=language-name	
/PACKAGE=package-name	

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and package associated with each specified parameter. If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes LSE to display all the information available about each specified parameter, as specified by the current DEFINE PARAMETER command (see the list of qualifiers for the DEFINEPARAMETER command).

If you specify an explicit name for the parameter, /FULL is the default.

/LANGUAGE=language-name

Shows only those parameters associated with the specified language. If you do not specify a language, LSE uses the current language. If you specify/LANGUAGE=*, LSE displays information on any parameter that matches the parameter name, regardless of the language for which it is defined.

The /LANGUAGE qualifier is mutually exclusive with the /PACKAGE qualifier.

/PACKAGE=package-name

Specifies the name of the package with which the parameter is associated. The/PACKAGE qualifier is mutually exclusive with the /LANGUAGE qualifier.

Parameter

parameter-name

Specifies which parameters are to be shown. If you omit this name, LSE assumes you have specified a wildcard parameter name.

Description

The SHOW PARAMETER command displays the definitions and characteristics of one or more parameters.

Related Commands

DEFINE PARAMETER

Example

```
LSE> SHOW PARAMETER id
```

Displays all the characteristics defined for the parameter *id*.

SHOW PLACEHOLDER

SHOW PLACEHOLDER — Displays the characteristics of the specified placeholders.

Format

```
SHOW PLACEHOLDER [placeholder-name]
```

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text
/LANGUAGE=language-name	

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and description of each placeholder.

If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

The /FULL qualifier causes LSE to display all the information available about each specified placeholder, as specified by the current DEFINE PLACEHOLDER command (see the list of qualifiers for the DEFINE PLACEHOLDER command).

If you specify an explicit name for the parameter, /FULL is the default.

/LANGUAGE=language-name

Associates a language with the specified placeholders. If you do not specify a language, LSE associates placeholders with the current language. If you specify /LANGUAGE=*, LSE displays information on any placeholder that matches the placeholder name, regardless of the language for which it is defined.

Parameter

placeholder-name

Specifies which placeholders are to be shown. If you omit this parameter, LSE assumes you have specified a wildcard placeholder name.

Description

The SHOW PLACEHOLDER command displays the definitions and characteristics of one or more placeholders.

Related Commands

DEFINE PLACEHOLDER

Example

```
LSE> SHOW PLACEHOLDER parameter
```

Displays all the characteristics defined for the placeholder parameter.

SHOW QUERY

SHOW QUERY — Displays information about one or more current SCA query sessions.

Format

SHOW QUERY [query-name,...]

Qualifiers	Defaults
/BRIEF	/BRIEF
/FULL	/BRIEF

Qualifiers

/BRIEF (D)

Indicates how much information you want to be displayed. The /BRIEF qualifier causes SCA to display (in tabular format)the query name, query expression, and description for the specified query.

/FULL

Indicates how much information you want to be displayed. The /FULL qualifier causes SCA to display all information about the specified query.

Parameter

query-name

Specifies the name of the query to be displayed. If you specify a null query name (" "), SCA assumes you mean the current query. If you omit this parameter, SCA assumes you have specified an asterisk (*).

Related Commands

FIND

GOTO QUERY

Example

```
LSE> SHOW QUERY
```

Displays one line of information on all current SCA queries.

SHOW ROUTINE

SHOW ROUTINE — Displays the characteristics of one or more routines.

Format

SHOW ROUTINE [routine-name]

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text
/LANGUAGE=language-name	
/PACKAGE=package-name	

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and package associated with each routine.

If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes LSE to display all the information available about each specified routine, as specified by the current DEFINE ROUTINE command (see the list of qualifiers for the DEFINE ROUTINE command).

If you specify an explicit name for the parameter, /FULL is the default.

/LANGUAGE=language-name

Shows routines that are associated with the specified language. If you do not specify a language, LSE uses the current language. If you specify/LANGUAGE=*, LSE displays information on any routine

that matches the routine name, regardless of the language for which it is defined. The /LANGUAGE and/PACKAGE qualifiers are mutually exclusive.

/PACKAGE=package-name

Specifies the name of the package with which the routine is associated. The /PACKAGE and /LANGUAGE qualifiers are mutually exclusive.

Parameter

routine-name

Indicates which routines are to be displayed. If you omit this parameter, LSE assumes you have specified a wildcard routine name.

Description

The SHOW ROUTINE command displays the definitions and characteristics of one or more routines.

Related Commands

DEFINE ROUTINE

Example

```
LSE> SHOW ROUTINE sys$add_holder
```

Displays all the characteristics defined for the routine sys$add_holder.

SHOW SCREEN

SHOW SCREEN — Displays the current values set with the SET SCREEN command.

Format

SHOW SCREEN

Description

The SHOW SCREEN command displays the current values for keywords used with the SET SCREEN command.

For users of the DECwindows interface, the SHOW SCREEN command uses the Window Attributes dialog box to display the screen attributes. This dialog box permits you to change the screen settings.

DECwindows Interface Equivalent

Pull-down menu: Options → Window Attributes . . .

Related Commands

SET SCREEN

Example

```
LSE> SHOW SCREEN
```

Displays all the screen attributes set by the WIDTH, HEIGHT,WINDOW, BALANCE_WINDOWS, and MINIMUM_WINDOW_LENGTH keywords of the SET SCREEN command, and the fonts set by the SET FONT command.

SHOW SEARCH

SHOW SEARCH — Displays the settings of text-search options.

Format

SHOW SEARCH

Description

The SHOW SEARCH command shows the current settings of the various text-search options. In DECwindows mode, LSE uses the Search Attributes dialog box to display the search settings. This dialog box permits you to change the settings.

DECwindows Interface Equivalent

Pull-down menu: Options > Search Attributes …

Related Commands

SEARCH

SET SEARCH

SHOW SOURCE_DIRECTORY

SHOW SOURCE_DIRECTORY — Displays the setting of the SET SOURCE_DIRECTORY command.

Format

SHOW SOURCE_DIRECTORY

Description

The SHOW SOURCE_DIRECTORY command displays the list of directories specified by the SET SOURCE_DIRECTORY command.

Related Commands

SET SOURCE DIRECTORY

SHOW SUMMARY

SHOW SUMMARY — Shows statistics and other information about LSE.

Format

SHOW SUMMARY

Description

The SHOW SUMMARY command shows statistics and other information about LSE, as follows:

- Version number of the software
- Current journal file specification (if any)
- Current section file specification
- Total number of buffers (system- and user-created)
- Modules used in the section file
- Other information about the LSE configuration

This information is useful for DECTPU programming, or in case you need to submit a software performance report (SPR).

To scroll through the list, use the Next Screen and Prev Screen keys. To return to the buffer you were editing, press the Return key.

SHOW TAG

SHOW TAG — Displays the characteristics of the specified tags.

Format

SHOW TAG [tag-name]

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text
/LANGUAGE=language-name	

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and description of each tag.

If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

The /FULL qualifier causes LSE to display all the information available about each specified tag, as specified by the current DEFINE TAG command (see the list of qualifiers for the DEFINE TAG command).

If you specify an explicit name for the parameter, /FULL is the default.

/LANGUAGE=language-name

Associates a language with the specified tags. If you do not specify a language, LSE associates tags with the current language. If you specify /LANGUAGE=*, LSE displays information on any tag that matches the tag name, regardless of the language for which it is defined.

Parameter

tag-name

Specifies which tags are to be shown. If you omit this parameter, LSE assumes you have specified a wildcard tag name.

Description

The SHOW TAG command displays the definitions and characteristics of tags.

Related Commands

DEFINE TAG

DELETE TAG

EXTRACT TAG

Example

```
LSE> SHOW TAG parameter
```

Displays all the characteristics defined for the tag parameter.

SHOW TOKEN

SHOW TOKEN — Displays the characteristics of one or more tokens.

Format

SHOW TOKEN [token-name]

Qualifiers	Defaults
/BRIEF	See text
/FULL	See text
/LANGUAGE=language-name	

Qualifiers

/BRIEF

Indicates how much information you want displayed. The /BRIEF qualifier causes LSE to display (in tabular format) the name and description of each token.

If you specify a wildcard expression for the parameter, or if LSE assumes one, /BRIEF is the default.

/FULL

Indicates how much information you want displayed. The /FULL qualifier causes LSE to display all the information available about each specified token, as specified by the current DEFINE TOKEN command (see the list of qualifiers for the DEFINE TOKEN command).

If you specify an explicit name for the parameter, /FULL is the default.

/LANGUAGE=language-name

Associates a language with the specified tokens. If you do not specify a language, LSE associates tokens with the current language. If you specify/LANGUAGE=*, LSE displays any tokens that match the token name,regardless of the language for which it is defined.

Parameter

token-name

Indicates which tokens are to be displayed. If you omit this parameter, LSE assumes you have specified a wildcard token name.

Description

The SHOW TOKEN command displays the definitions and characteristics of one or more tokens.

Related Commands

DEFINE TOKEN

Example

```
LSE> SHOW TOKEN/LANGUAGE=EXAMPLE ASSIGNMENT
```

Displays the characteristics defined for the token ASSIGNMENT associated with the language EXAMPLE.

SHOW VERSION

SHOW VERSION — Displays the current version of LSE and SCA.

Format

SHOW VERSION

Description

The SHOW VERSION command displays the current version of LSE and SCA.

If you are using SCA alone, only the SCA version is displayed. If you are using LSE, the LSE version is displayed, and the SCA version is displayed if SCA is installed on your system.

Examples

1. `$ SCA SHOW VERSION%SCA-S-VERSION, this is SCA version V4.6$`

   Displays the version of SCA that you are using.

2. ```
   LSE> SHOW VERSION
   This is LSE version V4.6
   This is SCA version V4.6
   LSE>
   ```

   Displays the version of LSE and SCA that you are using.

SHRINK WINDOW

SHRINK WINDOW — Shrinks the current window.

Format

SHRINK WINDOW line-count

Parameter

line-count

Indicates the number of screen lines you want to subtract from the current window. The maximum size of a window depends on the size and type of the terminal screen you are using. The minimum size is one line of text and one line for the status line.

Description

The SHRINK WINDOW command shrinks the window that the text cursor is in (if you are using more than one window). LSE enlarges the other window (or windows) accordingly.

Related Commands

ENLARGE WINDOW

Example

```
LSE> SHRINK WINDOW 5
```

Subtracts five lines from the current window and apportions the lines to the other windows you have on the screen.
```

SPAWN

SPAWN — Spawns a subprocess running the DCL command interpreter and suspends the editing session. Note, that this function is not available in DECwindows; any attempt to invoke it incurs an error.

Format

SPAWN [command]

Parameter

command

Specifies a command line to be executed by the spawned subprocess. If you specify this parameter, the subprocess ends and LSE regains control upon completion of the command.

Description

The SPAWN command suspends the current LSE session and connects your terminal to a new OpenVMS process at the DCL level. To resume your editing session, logout of the OpenVMS process, or use the DCL command ATTACH to resume the editor process.

This command is useful for running screen-oriented programs and OpenVMS utilities without ending the current editing session.

Related Command

ATTACH

Example

```
LSE> SPAWN
```

Connects you to a new subprocess. The DCL dollar sign ($) prompt signifies subprocess connection.

SPELL

SPELL — Runs DECspell to check the currently selected text or the entire buffer.

Format

SPELL

Description

The SPELL command runs DECspell (if it is installed on your system) to check the currently selected text or the entire buffer.

Use the following steps:

1. Select the text you want to check. If you do not select any text, SPELL checks the entire buffer.

2. Enter the SPELL command. If you select less than a full line, LSE extends the selected range to include the beginning and end of the line containing the range.

If the selected range (or the entire buffer if you do not select any text) contains any overview records, a message informs you that the operation cannot be performed.

LSE spawns a subprocess to run DECspell and writes out the current buffer or selected range to a temporary file in SYS$SCRATCH. (The name of the temporary file uses the subprocess PID.)

When SPELL finishes, LSE replaces the buffer or selected range with the new version of the temporary file (with corrections) and deletes any old versions of the temporary file. You then resume editing.

Do not use Ctrl/Y with SPELL. Ctrl/Y deletes lines in the temporary output file, which destroys the selected range or current buffer.

You use the SPELL command only with DECwindows.

SPLIT WINDOW

SPLIT WINDOW — Divides the current window into two or more windows.

Format

SPLIT WINDOW [window-count]

Parameter

window-count

Specifies the number of windows to create. The maximum size of a window depends on the size and type of the terminal screen you are using. The minimum size is one line of text and one line for the status line.

The text cursor appears in the lowest of the new windows.

Description

The SPLIT WINDOW command splits the current window into two or more windows. LSE displays the current buffer in each of the new windows.

DECwindows Interface Equivalent

SPLIT WINDOW 2

Pull-down menu: View → New Window

Related Commands

CHANGE WINDOW_MODE

DELETE WINDOW

Example

```
LSE> SPLIT WINDOW 4
```

Splits the current window into 4 windows with the current buffer displayed in each.

SUBSTITUTE

SUBSTITUTE — Replaces occurrences of one text string with another.

Format

SUBSTITUTE search-string

replace-string

Qualifiers	Defaults
/ALL	/CONFIRM
/[NO]CASE_MATCHING	/NOCASE_MATCHING
/CONFIRM	/CONFIRM
/DIALOG	/NODIALOG
/[NO]PATTERN	/NOPATTERN
/SINGLE	/CONFIRM

Qualifiers

/ALL

Specifies that all occurrences of the search string are to be replaced with the replace string. Specifying the /ALL qualifier causes LSE to perform all the specified substitutions without prompting you for further instructions.

/CASE_MATCHING
/NOCASE_MATCHING (D)

Specifies whether LSE uses the case of words in the search string to determine the case for the replacement string. The four conditions are:uppercase, lowercase, capitalized, or undetermined. For example, if a word in the search string is all uppercase, all the letters of the corresponding word in the replacement string become uppercase. If a word in the search string does not match the criteria for uppercase, lowercase, or capitalization, or there are no alphabetic characters in the search string word, its case is undetermined and LSE does not modify the case of the corresponding word in the replacement string.

If the replacement string contains more than one word, LSE respectively matches the case of words in the replacement string with the case of the corresponding words in the search string. If the search string contains fewer words than the replacement string, LSE matches the case of the additional words of the replacement string with the case of the last word in the search string.

Specifying the /NOCASE_MATCHING qualifier causes LSE not to modify the case of the replacement string to match that of the search string.

/CONFIRM (D)

Instructs LSE to prompt you for a confirmation at each occurrence before performing a substitution. If you specify the /CONFIRM qualifier, LSE highlights each occurrence of the search string located by the search and prompts you for an action. Enter one of the following responses:

- YES instructs LSE to replace this occurrence.
- NO instructs LSE not to replace this occurrence, but to proceed with the command.
- QUIT ends the command without replacing this occurrence and stops the SUBSTITUTE operation.
- ALL replaces this occurrence and all remaining occurrences without further prompting.

/DIALOG
/NODIALOG (D)

Instructs LSE to use a dialog box to prompt you for parameters and qualifier values. If you specify this qualifier, the command parameters are optional. If you supply command parameters and qualifiers with the /DIALOG qualifier, LSE uses those parameters and qualifiers to set the initial state of the dialog box.

The Substitute dialog box has the same fields as the Search dialog box, plus a button for case-matching replacement and a text field for the replacement string.

/PATTERN
/NOPATTERN (D)

Enables or disables special interpretation of wildcard characters and a quoting character in the *search-string* parameter. You can use the SET SEARCH command to set the syntax for specifying a pattern to either OpenVMS style, UNIX style or TPU style. For listing of OpenVMS- and UNIX-style wildcards, see the /PATTERN qualifier on the SEARCH command.

For more details on TPU patterns see Appendix G and *DEC Text Processing Utility Reference Manual*.

When the /NOPATTERN qualifier is specified (or is the default), special interpretation of the asterisk, percent sign, and backslash characters is disabled.

/SINGLE

Specifies that only one occurrence of the search string is to be replaced with the replacement string. Specifying the /SINGLE qualifier causes LSE to perform a single substitution without prompting you for an action.

Parameters

search-string

Specifies the string for which to search.

replace-string

Specifies the string to substitute.

Description

The SUBSTITUTE command replaces one string of text with another. If the *search-string* and *replace-string* parameters appear on the command line, you should enclose each string in quotation marks. To obtain expected results, this is required if the search string contains (or you want the replacement string to contain) lowercase or non-alphanumeric characters.

If LSE prompts you for search and replace strings, you must omit any quotation marks that are not part of the text of the string.

LSE performs the search in the current direction. If you specify a null string for a search string, LSE uses the last search string specified in a SEARCH or SUBSTITUTE command. The SUBSTITUTE command differs from the SEARCH command in that, with the SUBSTITUTE command, LSE does not ignore an occurrence of the search string at the current cursor position.

When the substitution is complete, LSE leaves the cursor at the end of the last changed occurrence.

If you specify a repeat count, LSE ignores the count unless you specify the/SINGLE qualifier.

If the cursor is beyond the target of the search, LSE displays a message in the message buffer informing you that the target was not found.

DECwindows Interface Equivalent

SUBSTITUTE/DIALOG

Pull-down menu: Search → Substitute . . .

Related Commands

SEARCH

SET SEARCH

Examples

1. `LSE> SUBSTITUTE "man" "person"`

   Moves the cursor to the first occurrence of the word *man* in the current direction and invokes the confirmation prompt. A positive response replaces the word *man* with the word *person*.

2. ```
   LSE>  SUBSTITUTE/CASE_MATCHING
   _Search for:  str$append
   _Replace with:  str$prefix
   ```

   Moves the cursor to the first occurrence of the string *str$append* in the current direction. A positive response to the confirmation replaces *str$append* with *str$prefix*. If *str$append* occurs in uppercase(STR$APPEND), LSE puts the replacement string in uppercase (STR$PREFIX) even though you specified it in lowercase on the command line.

3. `LSE> SUBSTITUTE/PATTERN "NAME_%_LENGTH" "NAME_B_LENGTH"`

   Moves the cursor to the next occurrence of a string consisting of NAME_ and _LENGTH separated by any single character. A positive response to the confirmation prompt replaces that string with the string NAME_B_LENGTH.
   ```

TAB

TAB — Inserts indentation. If the cursor is at the beginning of the line, it moves to the current indentation level; otherwise, the cursor moves to the next tab stop.

Format

TAB

Description

The TAB command inserts blanks and tabs to move the cursor to the current indentation level (if at the beginning of the line), or to move the cursor to the next tab stop as set by the /TAB_INCREMENT qualifier on the DEFINE LANGUAGE command or by the SET TAB_INCREMENT command.

If the current indentation level is set to the beginning of the line and the cursor is at the beginning of the line, the TAB command inserts enough blank space to move the cursor to the first tab stop. In contrast, the ENTER TAB command has no effect when both the cursor and current indentation level are at the beginning of the line.

Keypad Equivalent

Key	Keypad Equivalent
Ctrl/I **TAB**	All

Related Commands

ENTER TAB

SET TAB_INCREMENT

UNTAB

TOGGLE SELECT_MARK

TOGGLE SELECT_MARK — Sets or cancels the SELECT_MARK state.

Format

TOGGLE SELECT_MARK

Description

The TOGGLE SELECT_MARK command sets the select mark if it is not set, and cancels the select mark if it is set.

Keypad Equivalent

Key	Keypad Equivalent
E4 **SELECT**	EDT LK201, EVE LK201

Key	Keypad Equivalent
KP7 **SELECT**	EVE VT100

DECwindows Interface Equivalent

Pull-down menu: Edit → Select_mark

Related Commands

CANCEL SELECT_MARK

SET SELECT_MARK

TWO WINDOWS

TWO WINDOWS — Splits the current window into two windows.

Format

TWO WINDOWS

Description

The TWO WINDOWS command splits the current window into two smaller windows. (This command is the same as the SPLIT WINDOW command except it does not take a parameter.) You can view different buffers at the same time, or different parts of the same buffer.

The cursor appears in the new lower window. Each window has its own status line and displays the buffer you are currently editing. To put a different buffer in the window, use one of the following commands:

GOTO BUFFER
GOTO FILE
NEXT BUFFER (if you have created more than one buffer)

To continue splitting windows, repeat the TWO WINDOWS command.

DECwindows Interface Equivalent

Pull-down menu: View → New Window

Related Commands

CHANGE WINDOW_MODE

DELETE WINDOW

ENLARGE WINDOW

ONE WINDOW

OTHER WINDOW

PREVIOUS WINDOW

SET SCREEN

SHRINK WINDOW

Example

```
LSE> TWO WINDOWS
```

Splits the current window into two windows.

UNDO

UNDO — Reverses the most recently executed LSE editing operation for the current buffer.

Format

UNDO

Description

The UNDO command is used to undo the previous operation on the current buffer. The command can be used repeatedly until there are no more operations to be undone, or until the maximum number of operations that can be undone for the buffer is reached.

Some operations (such as a call to an external TPU procedure) cannot be undone, and therefore cause undo information to be lost. An informational message is issued for each operation that cannot be undone.

The commands SET OVERSTRIKE and FOCUS result in the current undo information being lost for the appropriate buffer.

Operations that might result in user TPU code being executed result in the current undo information being lost for all buffers. For example:

```
DO/TPU/BUFFER=user.tpu
```

It should be noted that UNDO operations often do not have a one-to-one relationship with editing operations. For example,a series of cursor-positioning operations are treated as a single operation moving from the first position to the last position.

DECwindows Interface Equivalent

Pull-down menu: Edit → Undo

Related Command

REDO

SET MAX_UNDO

```
SET MODE UNDO=OFF

SET MODE UNDO=ON

SHOW MAX_UNDO
```

UNDO ENTER COMMENT

UNDO ENTER COMMENT — Reverses the effect of the last ENTER COMMENT command.

Format

```
UNDO ENTER COMMENT
```

Description

The UNDO ENTER COMMENT command deletes the comments created from pseudocode with the ENTER COMMENT command and restores the text to the pseudocode placeholders.

Related Commands

```
ENTER COMMENT
```

UNERASE

UNERASE — Restores the text deleted by the corresponding ERASE command that you most recently executed.

Format

```
UNERASE [erase-option]
```

Parameter

erase-option

The following are valid options with the UNERASE command:

CHARACTER
LINE
PLACEHOLDER
SELECTION
WORD

Description

The UNERASE command restores text erased by the previous ERASE CHARACTER, ERASELINE, ERASE PLACEHOLDER, ERASE SELECTION, or ERASE WORD command. LSE inserts the restored text before the current cursor position, except for UNERASEPLACEHOLDER, which restores the text to its original position.

If you do not specify an erase option, LSE restores the text erased by the previous ERASE {CHARACTER, LINE, PLACEHOLDER, SELECTION, WORD} command,whichever was the most recent.

The UNERASE PLACEHOLDER command also restores the placeholders created by the ENTER PSEUDOCODE command and erased by the ERASEPLACEHOLDER command.

Keypad Equivalent

UNERASE

Key	Keypad Mode
PF1-E2 **INSERT HERE**	EVE LK201

UNERASE CHARACTER

Key	Keypad Mode
PF1-keypad comma (,) **UND C**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

UNERASE LINE

Key	Keypad Mode
PF1-PF4 **UND L**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

UNERASE PLACEHOLDER

Key	Keypad Mode
PF1-Ctrl/K	All

UNERASE SELECTION

Key	Keypad Mode
None	All

UNERASE WORD

Key	Keypad Mode
PF1-keypad minus (–) **UND W**	EDT LK201, EDT VT100, EVE LK201
None	EVE VT100

Related Commands

ERASE CHARACTER

ERASE LINE

ERASE PLACEHOLDER

ERASE SELECTION

ERASE WORD

Example

```
LSE> UNERASE CHARACTER
```

Retrieves the contents of the deleted-character buffer.

UNEXPAND

UNEXPAND — Reverses the effect of the EXPAND command.

Format

UNEXPAND

Description

For LSE, the UNEXPAND command reverses the effect of the last EXPAND command. LSE deletes the range containing the text inserted as part of the last EXPAND command, and restores the token, placeholder, or alias that appeared at that position before the EXPAND command was entered.

Keypad Equivalent

Key	Keypad Equivalent
PF1-Ctrl/E	EDT LK201, EDT VT100
PF1-Ctrl//	EVE LK201, EVE VT100

Related Commands

EXPAND

UNRESERVE

UNRESERVE — Cancels the reservation of a CMS element with the same name and type as the input file for your current buffer.

Format

UNRESERVE

Description

The UNRESERVE command cancels the reservation in your current CMS library for an element with the same name and type as the input file for your current buffer. After successfully canceling a reservation, LSE deletes your current buffer and its corresponding file.

DECwindows Interface Equivalent

Pull-down menu: File → Unreserve

Related Commands

REPLACE

RESERVE

SET CMS

UNTAB

UNTAB — Erases blanks and tabs to the left of the cursor, which moves the cursor to the previous stop.

Format

UNTAB

Description

The UNTAB command removes blanks and tabs to the left of the cursor, which moves the cursor to the previous tab stop set by the /TAB_INCREMENT qualifier on the DEFINE LANGUAGE command, or by the SET TAB_INCREMENT command.

If no tabs or blanks immediately precede the cursor, this command has no effect. If nonblank or nontab characters are present in the column positions at or after the previous tab stop, LSE removes the blanks and tabs between those characters and the cursor, then repositions the cursor after those characters, not at the tab stop.

Keypad Equivalent

Key	Keypad Equivalent
PF1-TAB	All

Related Commands

ENTER TAB

TAB

UPPERCASE WORD

UPPERCASE WORD — Changes the current word to uppercase.

Format

UPPERCASE WORD

Description

The UPPERCASE WORD command puts the current word in uppercase letters. If the word is in both lowercase and uppercase letters, LSE changes all letters to uppercase.

If the cursor is between words, LSE puts the following word in uppercase letters. If a selected range is active, all the words within that range are changed to uppercase. Then, the cursor moves to the start of the next word.

DECwindows Interface Equivalent

Pull-down menu: Edit → Uppercase

Related Commands

CHANGE CASE

LOWERCASE WORD

VERIFY

VERIFY — Verifies that the specified SCA libraries are valid, and repairs any corruptedlibraries.

Format

VERIFY [library-spec[, . . .]]

Qualifiers	Defaults
/[NO]LOG	/LOG
/[NO]RECOVER	/NORECOVER

Qualifiers

/LOG (D)
/NOLOG

Indicates whether SCA reports the library verification or repair operation.

/RECOVER
/NORECOVER (D)

Indicates whether SCA should repair a corrupted library.

If the interrupted command is a LOAD command, SCA deletes from the library any module that had begun to load but had not completed loading. Also, SCA cannot recover modules that were waiting to be processed for loading when the interruption occurred. To load interrupted and waiting modules, enter a subsequent LOAD command and include those modules.

If the interrupted library operation is a DELETE MODULE command, the /RECOVER qualifier causes SCA to delete the incompletely deleted module. Any modules still waiting to be processed for deletion when the interruption occurred are excluded from the recovery operation; to delete them, you must respecify them in a subsequent DELETE MODULE command.

Parameter

library-spec[, . . .]

Specifies the SCA libraries to be verified. If you do not specify a library, SCA assumes you have specified the primary library.

Description

The VERIFY command performs the following operations to verify the validity of specific SCA libraries:

- Checks for corrupted libraries resulting from abnormal termination of a LOAD or DELETE MODULE command
- Optionally, repairs corrupted libraries

Example

```
SCA> VERIFY/RECOVER SCA$:[USER.SCA]
```

Determines whether the library SCA$:[USER.SCA] has been corrupted and repairs any damage detected.

VIEW SOURCE

VIEW SOURCE — Displays an overview of the buffer.

Format

VIEW SOURCE

Qualifiers	Defaults
/DEBUG	
/DEPTH= *n*	/DEPTH=1

Qualifiers

/DEBUG

Provides a way to debug adjustment definitions by generating a copy of the source buffer, indented as LSE views the indentation. LSE displays the result in a system buffer named $OVERVIEW with all source lines visible. Numeric values for the indentation are also displayed with information about the adjustment applied to each line.

You cannot specify the /DEBUG qualifier with the /DEPTH qualifier.

/DEPTH=n
/DEPTH=1 (D)

Displays the top `n` levels of detail of the buffer. Lower levels are collapsed and represented by overview lines. If you specify /DEPTH=ALL, all the lines in the buffer are displayed; none of the lines are replaced by overview lines.

You cannot specify the /DEPTH qualifier with the /DEBUG qualifier.

Description

The VIEW SOURCE command displays the top `n` levels of detail of the entire buffer.

The editor determines the relative level of detail of a line by comparing the indentation of the line with the indentation of other lines. The editor's treatment of the indentation of a line is influenced by indentation-adjustment definitions. For more information,see the DEFINE ADJUSTMENT command.

Keypad Equivalent

VIEW SOURCE/DEPTH=1

Key	Keypad Mode
PF1- >	All

DECwindows Interface Equivalent

VIEW SOURCE/DEPTH=1

Pull-down menu: View → Overview Source

VIEW SOURCE/DEPTH=ALL

Pull-down menu: View → View Source

Related Commands

COLLAPSE

DEFINE ADJUSTMENT

DEFINE LANGUAGE/OVERVIEW_OPTIONS

EXPAND

FOCUS

MODIFY LANGUAGE

SET NOOVERVIEW

SET OVERVIEW

WHAT LINE

WHAT LINE — Shows the current line number and total number of lines in the buffer. Also shows what percentage of the lines in the buffer are located above the current line.

Format

WHAT LINE

Description

The WHAT LINE command shows the current line number and total number of lines in the buffer. It also shows what percentage of the lines in the buffer are located above the current line.

This command is useful if you want to know whether to insert a page break, or find out how many lines are in the buffer.

To move to a specific line by number, use the LINE command.

Related Commands

`LINE`

WRITE

WRITE — Writes the contents of a buffer, or the contents of the selected range, to a file.

Format

`WRITE [file-spec]`

Qualifiers	Defaults
/BUFFER=buffer-name	
/DIALOG	
/SELECT_RANGE	
/VISIBLE	

Qualifiers

/BUFFER=buffer-name

Indicates which buffer is to be written. The default is the current buffer.

/DIALOG

Instructs LSE to use a dialog box to prompt you for a parameter value. The command parameter is optional if you supply this qualifier. If you specify a command parameter with /DIALOG, LSE uses that parameter to set the initial state of the dialog box.

/SELECT_RANGE

Indicates that the selected range is to be written.

/VISIBLE

Indicates that the visible records in the buffer or selected range be written to a file. You must specify the *file-spec*parameter when you use this qualifier.

Parameter

file-spec

Specifies the file to which the buffer will be written. By default, LSE writes the data to the file associated with the buffer. This parameter is required if you specify the /SELECT_RANGE qualifier.

Description

The WRITE command places the contents of the specified buffer in the file you specify. Your editing session continues until you enter an EXIT or QUIT command. If you are editing an existing file and do not supply a new file name, LSE creates a new version of that file when you enter the WRITE command.

When you enter a WRITE command without specifying a file name, LSE also displays an informational message and prompts you for confirmation before writing the buffer under either of the following conditions:

- If you have not modified the buffer (not made any changes during your editing session)
- If the buffer's status is read-only

If you enter the WRITE command and the current buffer is associated with a file of the same name, LSE creates a new version of the file. If the buffer is unnamed, LSE prompts you for a name.

You can use the WRITE command and supply a file name at any time while you are in an editing session, which creates a new file containing the output up to that point in your editing session. However, using the WRITE command to write the data to a different file does not change the file association of the buffer; that is, LSE still creates a new version of the file with the same name as that associated with the buffer when you exit from that editing session, or subsequently use the WRITE command without specifying a file name. To change the file association, use the SET OUTPUT_FILE command.

If you use the WRITE command to write to a directory that you have set read-only (using the SET DIRECTORY command), LSE prompts you for confirmation before writing out the buffer.

DECwindows Interface Equivalent

WRITE

Pop-up menu: User buffer → Save

Pull-down menu: File → Save File

WRITE/DIALOG

Pull-down menu: File → Save As . . .

Related Commands

GOTO FILE

READ

SET OUTPUT_FILE

Example

```
LSE> WRITE/BUFFER=$SHOW SHOW.TXT
```

Causes LSE to write the current contents of the $SHOW buffer to a file called SHOW.TXT.

Appendix A. Interfacing to DECTPU Procedures

Some LSE commands depend on procedures written in the DECTPU programming language that are present in the LSE default section file(LSE$SECTION.TPU$SECTION). These procedures must be present for LSE to function properly. For this reason, if you want to use your own DECTPU section file, you must build it using LSE$SECTION.TPU$SECTION as a base. To do this successfully, your DECTPU procedures must obey certain rules described in this appendix.

VSI reserves all variable names and buffer names containing the dollar sign ($) character. You must not use names containing a dollar sign ($) in your own DECTPU code except as explained in the following sections.

A.1. DECTPU Variables and Procedures

The following three variable names have special meaning to LSE and DECTPU:

- MESSAGE_BUFFER—The buffer to which LSE writes messages
- SHOW_BUFFER—The buffer to receive output from the DECTPU SHOW built-in
- INFO_WINDOW—The window to which the DECTPU SHOW built-in maps SHOW_BUFFER

Your section file must not redefine TPU$INITIALIZE, the DECTPU procedure that LSE calls to set up the editing environment. LSE$SECTION.TPU$SECTION provides its own TPU$INIT_PROCEDURE. Instead, you should redefine TPU$LOCAL_INIT to perform initialization at startup time as described later in this appendix.

LSE uses the following DECTPU variables and procedures.

[TPU$LOCAL_INIT]

The LSE TPU$INIT_PROCEDURE calls this procedure after it has finished LSE initialization. LSE initialization includes processing the/INITIALIZATION qualifier and reading into a buffer the input file specified on the LSEDIT command. You can supply your own TPU$LOCAL_INIT procedure to initialize your own DECTPU variables and procedures.

[LSE$CREATE_SELECT_RANGE]

This procedure sets LSE$SELECT_RANGE through the following process. If LSE $START_SELECT_MARK is nonzero, it sets LSE$SELECT_RANGE to the range from LSE $START_SELECT_MARK to the current position, then zeros out LSE$START_SELECT_MARK and LSE$SELECT_IN_PROGRESS. Otherwise, if the cursor is positioned to the last string for which the user searched, it sets LSE$SELECT_RANGE to be a range containing that string. Otherwise, it sets LSE $SELECT_RANGE to 0.

[LSE$SET_STATUS_LINE (window)]

LSE calls this procedure whenever it wants to update the status line of a window. The procedure takes one argument – the window whose status line is to be set. If you want to change the status line for LSE, see the *VAX Text Processing Utility Manual*.

[LSE$MESSAGE_WINDOW]

This is a procedure that returns the window to which LSE maps MESSAGE_BUFFER.

[LSE$NUMBER_OF_WINDOWS]

This is a procedure that returns the number of windows mapped to the screen. Note that the number of windows may be more than two, which is the maximum for earlier versions of LSE.

[LSE$MAIN_WINDOW]

This procedure returns the top window displayed on the screen. It is compatible with earlier versions of LSE in which it was a variable that returned the window that was used in one-window mode. The current multiwindow implementation based on EVE creates and deletes windows as needed, making a backwards-compatible implementation of LSE$MAIN_WINDOW impossible.

[LSE$TOP_WINDOW]

These procedures return the top and bottom windows currently being displayed. They are compatible with earlier versions of LSE in which they were variables that returned the windows that were used in two-window mode.

[LSE$MAIN_BUFFER]

After LSE startup, this procedure points to the buffer containing the input file that appeared on the LSEDIT command line. When you exit from LSE,LSE remembers the current cursor position in this buffer. It is compatible with earlier versions of LSE in which it was a variable. It has been replaced by the variable EVE$X_MAIN_BUFFER.

[LSE$START_SELECT_MARK]

This procedure returns the contents of the EVE$X_SELECT_POSITION variable. It is compatible with earlier versions of LSE in which it was a variable. The contents of EVE$SELECT_POSITION can be either the select marker set by the SET SELECT_MARK command, or a range created by the SELECT ALL command or by using the mouse. If there is no SELECT operation in progress, its value is the integer 0.

[LSE$SELECT_IN_PROGRESS]

This procedure returns 1 if there is a SELECT operation in progress. If no SELECT operation is in progress, it returns 0.The value returned is computed by the TPU expression (EVE $X_SELECT_POSITION <> 0).

[LSE$SELECT_RANGE]

This is the range variable in which LSE$CREATE_SELECT_RANGE returns its value. LSE commands that act on the selected range use this variable.

Sample DECTPU Procedure

The following is a DECTPU procedure that demonstrates the use of an LSE selected range from a user-defined DECTPU procedure. Note the use of the variables LSE$SELECT_IN_PROGRESS and LSE $SELECT_RANGE, and the procedure LSE$CREATE_SELECT_RANGE.

```
PROCEDURE sort (qual)
  ! Description:
  !   Sorts the lines in the selected range.  Complete lines should be
  !   selected.  If no qualifiers are specified, the lines in the
```

```
  !   selected range are sorted in ascending order.
  !
  ! Parameter:
  !   qual - a string beginning with "$". The remainder of the string
  !     contains qualifiers to be passed to the SORT command.  The "$" is
  !     a dummy character.  It is there to serve as a parameter when no
  !     SORT qualifiers are specified, and to prevent qualifiers for
  !     SORT from being interpreted as qualifiers on the LSE CALL
  !     command.
  !
  LOCAL sort_process,cmd,save_position,current_message;
  ! If there is a selected range, write it to a temporary file.
  IF NOT LSE$SELECT_IN_PROGRESS
  THEN
      MESSAGE ('No select active');
      RETURN;
  ENDIF;
  LSE$CREATE_SELECT_RANGE;
  WRITE_FILE (LSE$SELECT_RANGE, 'sort_input.dat');
  ! Create a subprocess in which to run SORT.  Note that terminal output
  ! from the subprocess goes to the message buffer.
  sort_process := CREATE_PROCESS (message_buffer, 'SET NOON');
  ! Build the SORT command, picking up qualifiers that were passed in.
  cmd := 'SORT/STABLE sort_input sort_output
  '+SUBSTR(qual,2,LENGTH(qual)-1);
  ! Display the SORT command in the message window.
  MESSAGE (cmd);
  ! Execute the SORT command in the subprocess.
  SEND (cmd, sort_process);
  ! If no messages were written to the message buffer by SORT,
  ! assume that the SORT operation succeeded and replaced the
  selected range
  ! with the output from SORT.
  save_position := MARK(NONE);
  POSITION(message_buffer);
  current_message := CURRENT_LINE;
  POSITION (save_position);
  IF current_message = cmd
  THEN
      ERASE (LSE$SELECT_RANGE);
      READ_FILE ('sort_output.dat');
  ENDIF;
  ! Cleanup
  DELETE (LSE$SELECT_RANGE);
  SEND ('DELETE sort_input.dat;,sort_output.dat;',sort_process);
  DELETE (sort_process);
ENDPROCEDURE
```

To use the preceding procedure, define a SORT command as follows:

```
LSE> DEFINE COMMAND SORT "CALL SORT $"
```

To sort the lines in the selected range in ascending order, enter the following command:

```
LSE> SORT
```

To sort the lines in the selected range in descending order based on the text that begins in the 10th column and extends to, but does not include, the 20th column, enter the following command:

```
LSE> SORT/KEY=(POSITION=10, SIZE=10, DESCENDING)
```

A.2. Guidelines for User-Written TPU Procedures

You can transport user-written TPU procedures from EVE to LSE. Therefore, you can use code that calls EVE procedures within LSE.

The following LSE variables are now procedures. If a TPU procedure accesses the value of the corresponding variable, but does not assign a value to it, the TPU procedure should continue to work. If a TPU procedure must change the value of one of the following variables, you should change the TPU code to use the corresponding EVE variable, if any, shown in parentheses.

- LSE$MESSAGE_WINDOW (MESSAGE_WINDOW)
- LSE$NUMBER_OF_WINDOWS (EVE$X_NUMBER_OF_WINDOWS)
- LSE$MAIN_WINDOW
- LSE$TOP_WINDOW
- LSE$BOTTOM_WINDOW
- LSE$MAIN_BUFFER (EVE$X_MAIN_BUFFER)
- LSE$START_SELECT_MARK (EVE$X_SELECT_POSITION)
- LSE$SELECT_IN_PROGRESS (returns EVE$X_SELECT_POSITION <> 0)

To tailor window-status lines, see the information on EVE status-line fields in the *VAX Text Processing Utility Manual*.

A.2.1. Adding User-Written TPU Procedures

You can add user-written TPU procedures to LSE with the DECTPU tool EVE$BUILD. You use EVE$BUILD for modifying or adding user-written TPU procedures to LSE. EVE$BUILD compiles DECTPU code with an existing LSE section file to produce a new section file. See the *DEC Text Processing Utility Manual* for more information on using EVE$BUILD.

To extend LSE with EVE$BUILD, do the following:

1. Create a file called USER_MASTER.FILE that lists the files being used to extend LSE. For example:

   ```
   sys$login:abbreviation.tpu
   lseplus:auto_indent.tpu
   sys$login:customizations.tpu
   ```

2. Create a file called USER_VERSION.DAT that contains the version number to be associated with this section file, for example, Version 1.0.

3. Define a foreign command to use for builds. For example:

   ```
   $ BUILD == "LSEDIT/NODISP/NOINIT/COMM=SYS$EXAMPLES:EVE$BUILD"
   ```

4. Enter the command that builds this module in with the existing LSE section file. You will get messages that the definitions of various EVE procedures are being superseded, which you may ignore. For example:

```
$ BUILD USER
%TPU-S-FILEIN, xxx lines read from file SYS$EXAMPLES:EVE$BUILD.TPU
Definition of procedure EVE$BUILD_MODULE_INDENT superceded
.
.
.
Definition of procedure EVE$BUILD superceded
Section file name [default = product name USER]:
```

   At this point, you must enter the name and location of the section file that you want to create.

5. Press the Return key to create a section file named USER.TPU$SECTION in the current directory.

To use this newly created section file, invoke LSE with the /SECTION qualifier and supply the full file specification that corresponds to the section file. Alternatively, you can define the logical name LSE $SECTION to be the full file specification for this new section file.

A.2.2. DECTPU Programming with Hidden Records in LSE

With LSE, you use the COLLAPSE, EXPAND, FOCUS, and VIEW SOURCE commands for viewing source code. This code elision feature means that there can be four different types of records (or lines) in a buffer. A record can be a source record or an overview record, and can be visible or hidden.

Source records correspond to the actual text that is read from a file, edited, and written to a file. Overview records are inserted by LSE and are representatives for source records that have been hidden or omitted. Overview records themselves are hidden when the corresponding source is made visible. Overview records might also be hidden along with source records, such as when a set of lines containing both source lines and overview lines is collapsed to an overview.

Thus, the four types of records are as follows:

- Visible source record
- Hidden source record
- Visible overview record
- Hidden overview record

With TPU, the current position in a buffer can be on any one of these types of records. The TPU built-ins MOVE_VERTICAL and MOVE_HORIZONTAL move from record to record and are not influenced by the visibility or whether the record is an overview. A TPU procedure that does not consider visibility or overview records might not function as intended if you use the elision facility prior to calling the procedure.

After each LSE command, if the current position is not on a visible record, LSE makes the record visible. If the current position is on a hidden source record,LSE expands sufficient overviews to make the record visible. If the current position is on a hidden overview record, LSE collapses the source to make the overview visible.

Overview records are not modifiable. If you attempt to alter the text, split the record, or append the record to another record, it will fail.

A number of built-ins are available for you to enhance or develop TPU procedures to work when there are overviews or hidden records in a buffer. The built-ins are listed here and described in *Section A.3, "Supplemental DECTPU Built-Ins"*:

- LSE$IS_OVERVIEW
- LSE$IS_VISIBLE
- LSE$MAKE_VISIBLE
- LSE$MOVE_BY_SOURCE
- LSE$MOVE_HORIZONTAL
- LSE$MOVE_TEXT
- LSE$MOVE_VERTICAL
- LSE$NEAREST_VISIBLE
- LSE$SOURCE_ONLY

A.3. Supplemental DECTPU Built-Ins

LSE supports new DECTPU built-in procedures and extends some of the existing built-ins, as described in the following sections.

A.3.1. LSE$DO_COMMAND (String)

Takes a single character string as its argument. It executes the string as an LSE command. You can use this built-in to execute LSE commands from within your DECTPU procedures.

A.3.2. LSE$GET_ENVIRONMENT(String, Keyword)

Incorporates the definitions contained in an environment file into the editing session. There are two arguments, as follows:

- *string* — Specifies the file specification of the environment file.
- *keyword* — Specifies the keyword that indicates whether definitions from the file should be written out by the LSE SAVE ENVIRONMENT command. The possible keywords are as follows:
  - ON—Write out the definitions.
  - OFF—Do not write out the definitions.

A.3.3. GET_INFO (buffer, ˝language˝)

Returns a string representing the name of the language currently associated with the given buffer. If there is no language associated with the buffer, the integer 0 is returned.

A.3.4. GET_INFO (buffer, ˝overviews˝)

Returns the keyword ON or OFF, based on whether overview operations are allowed in the given buffer.

A.3.5. GET_INFO(COMMAND_LINE, item)

LSE provides the following additional COMMAND_LINE items for the GET_INFO built-in:

- CHARACTER Returns an integer containing the starting character position in the starting line for the edit. The first character position in the line is character 1. This is the value from the second number in the /START_POSITION qualifier,or the value from translating the logical name LSE $START_CHARACTER. This item is a synonym for the START_CHARACTER item maintained for compatibility with earlier versions of LSE.
- ENVIRONMENT Returns 1 if the /ENVIRONMENT qualifier is present on the command line; otherwise, it returns 0.
- ENVIRONMENT_FILE Returns a string containing the file specification from the / ENVIRONMENT qualifier. The /ENVIRONMENT qualifier specifies a list of file specifications. Each time a GET_INFO(COMMAND_LINE, “ENVIRONMENT_FILE”) built-in call is done, LSE returns the next file specification in the list. It returns the null string on all calls after the end of the list is reached.

  This built-in call returns the null string if /ENVIRONMENT was not present on the command line.
- LANGUAGE Returns a string containing the language name from the/LANGUAGE qualifier on the command line, or the null string if /LANGUAGE was not specified.
- LINE Returns an integer containing the starting line number for LSE. The first line in the file is considered line 1.This is the value from the /START_POSITION qualifier, or the translation of the logical name LSE$START_LINE. This item is a synonym for the START_RECORD item maintained for compatibility with earlier versions of LSE.
- SYSTEM_ENVIRONMENT Returns 1 if the /SYSTEM_ENVIRONMENT qualifier is present on the command line; otherwise, it returns 0.
- SYSTEM_ENVIRONMENT_FILE Returns a string containing the file specification from the / SYSTEM_ENVIRONMENT qualifier, or it returns the null string if /SYSTEM_ENVIRONMENT is not present on the command line.
- CURRENT_FILE Returns 0 if the /NOCURRENT_FILE qualifier is specified on the command line, and returns 1 if the /CURRENT_FILE qualifier is specified on the command line.

A.3.6. LSE$FIND_OPEN_COMMENT (marker)

Returns a range that corresponds to the first open-comment delimiter found after the marker, but on the same line as the marker.

Returns 0 if there is no language associated with the buffer containing the marker.

A.3.7. LSE$FIND_CLOSE_COMMENT (marker)

Returns a range that corresponds to the first close-comment delimiter found after the marker, but on the same line as the marker.

Returns 0 if there is no language associated with the buffer, or if no close comment is found.

A.3.8. LSE$IS_OVERVIEW [(marker)]

Returns 1 if the indicated record is an overview record and 0 if it is a source record. If the marker parameter is not specified, the current record is used.

A.3.9. LSE$IS_VISIBLE [(marker)]

Returns 1 if the indicated record is a visible record and 0 if it is a hidden record. If the marker parameter is not specified, the current record is used.

A.3.10. LSE$MOVE_HORIZONTAL (integer)

Restricts the cursor to visible records. LSE does not count the characters or end-of-line on hidden lines when determining where to establish the new editing point. If the original editing point is on a hidden record, the movement to a visible record counts as a move of one line. (Similar to the TPU MOVE_HORIZONTAL built-in.)

A.3.11. LSE$MOVE_VERTICAL (integer)

Restricts the cursor to visible records. Hidden records are not counted as they are traversed and the cursor cannot be left on a hidden record.

A.3.12. LSE$MOVE_BY_SOURCE (integer)

Restricts the cursor to source records only. LSE does not count overview lines when determining where to establish the new editing point. If the original editing point is on an overview line, the movement to a source line counts as a move of one line. If the source line onto which the cursor is to move is hidden, LSE$MOVE_BY_SOURCE makes the source line visible. (Similar to the TPU MOVE_VERTICAL built-in.)

A.3.13. LSE$MAKE_VISIBLE (marker |range)

Makes the specified records visible. If a marker is specified, LSE makes the corresponding record visible by expanding overview lines. If a range is specified, LSE makes all the source records in the range visible by expanding sufficient overviews.

A.3.14. LSE$NEAREST_VISIBLE (marker)

Moves the editing position to the beginning of the visible line nearest to the specified position. If the record at the specified position is visible, it moves the current editing position there. If the marker parameter is not specified, the current editing position is used. This is useful for operations that move the editing position to a new record but should not change the view,for example, moving the cursor by using a scroll bar, or moving the cursor to a window where the last position in that window has become hidden.

A.3.15. LSE$SOURCE_ONLY (range)

Returns 1 if all the source records within the range are visible; otherwise it returns 0.If all the source records within the range are visible, then, as a side effect,all the hidden overview records in the range are deleted. This function is useful when writing a TPU procedure that operates on a range. It does not

operate properly if there are hidden records or overview records in the range. For example, this built-in is used in the procedure that implements the FILL operation.

A.3.16. LSE$MOVE_TEXT and LSE$COPY_TEXT (string |range |buffer)

Move or copy the text from the specified string, range, or buffer to the current editing position and return the resultant range. If the input is a string, these functions are equivalent to MOVE_TEXT and COPY_TEXT. For ranges and buffers, the LSE functions preserve overview information. (Similar to the TPU built-ins MOVE_TEXT and COPY_TEXT.)

Overview information is language-dependent, so the language associated with the input range or buffer must be the same as the language associated with the current buffer. If the input language is not the same as the language for the current buffer, there are side effects, as follows:

- If the current buffer can legally accept overview records, and if the current buffer is empty, or if the current buffer has no associated language and contains no overview records, the current buffer inherits the language of the input buffer. A buffer containing no records or only one null record is considered empty.
- In the case of LSE$COPY_TEXT, only visible records are copied. Visible overview records in the result range are marked as source records. In this case, LSE$MOVE_TEXT aborts to avoid the loss of hidden source lines.

If LSE$MOVE_TEXT is given a range, any hidden overview lines immediately preceding the range are deleted.

If there are overview records in a range or buffer, the TPU functions MOVE_TEXT and COPY_TEXT change the overview records into source records. Visibility of records is preserved.

LSE$COPY_TEXT and LSE$MOVE_TEXT will not operate on an input range that includes part, but not all, of an overview line. An overview line includes the line break at its end.

A.3.17. SET (LSE$LANGUAGE, buffer, language)

Associates or disassociates a language and a buffer. See the descriptions of the SET LANGUAGE and SET NOLANGUAGE commands in this manual for a more complete discussion of associating a language with a buffer.

The arguments to the built-in are the keyword LSE$LANGUAGE, followed by a buffer variable, followed by the language string. The literal current buffer can be used as the buffer variable. The language string can be passed as double quotes(“”), which results in disassociating the language from the buffer.

A.3.18. SET (LSE$OVERVIEWS, buffer, on/off)

Enables or disables overview operations in the indicated buffer. See the descriptions of the SET OVERVIEW and SET NOOVERVIEW commands in this manual for a more complete description of overview operations.

The arguments to the built-in are the keyword LSE$OVERVIEWS, followed by a buffer variable, followed by either the keyword ON or the keyword OFF. You can use the literal current_buffer as the buffer variable.

A.3.19. TPU Built-ins for the SCA Callable Interface

There are TPU built-ins for the SCA$QUERY_xxx functions in the new SCA callable interface. Specifically, the built-ins are as follows:

- SCA$QUERY_CLEANUP
- SCA$QUERY_COPY
- SCA$QUERY_FIND
- SCA$QUERY_GET_ATTRIBUTE
- SCA$QUERY_GET_ATTRI_KIND_T
- SCA$QUERY_GET_ATTRI_VALUE_T
- SCA$QUERY_GET_OCCURRENCE
- SCA$QUERY_GET_NAME
- SCA$QUERY_INITIALIZE
- SCA$QUERY_PARSE
- SCA$QUERY_SELECT_ENTITY

None of the other routines are available. LSE calls SCA$INITIALIZE and SCA$CLEANUP automatically for you. The command context created by LSE is available in the TPU variable LSE$SCA_COMMAND_CONTEXT; you must use this as the first parameter to call SCA $QUERY_INITIALIZE.

SCA message codes are available as TPU keywords, as in the conventional format SCA$_xxx. These can be used as message constants for the TPU MESSAGE built-in.

SCA constants for the attribute kinds are available as TPU constants in the form SCA$K_ATTRI_xxx. These can be passed directly to SCA$QUERY_GET_ATTRI_VALUE_T.

LSE handles the calling sequences for you. You need not be concerned with whether objects are passed by value or reference.

Note that TPU does not produce a traceback if an SCA routine signals an error.

Appendix B. Language-Specific Information

This appendix contains information of interest to VSI Fortran and VSI COBOL programmers. *Section B.1, "VSI Fortran"* provides information on using VSI Fortran with LSE. *Section B.2, "VSI COBOL"*provides information on using VSI COBOL with LSE.

B.1. VSI Fortran

Some LSE commands behave differently when the definition of the current language includes the /FORTRAN qualifier. The syntax of this qualifier is as follows:

```
/FORTRAN=[NO]ANSI_FORMAT
```

ANSI_FORMAT specifies that templates should be expanded in ANSI format. The default is NOANSI_FORMAT (tab format).

To choose a format that is different from the format specified in the /FORTRAN qualifier on the language definition, use the MODIFY LANGUAGE command and specify /FORTRAN=ANSI_FORMAT or /FORTRAN=NOANSI_FORMAT.(See the MODIFY LANGUAGE command in the Command Dictionary for more information on the /FORTRAN=[NO]ANSI_FORMAT qualifier.)

VSI Fortran Source Format

The VSI Fortran compiler supports two source-line formats:ANSI format and tab format.

VSI Fortran differs from the other languages supported by LSE in that each source line is divided into three fields. These fields are as follows:

- Statement number field
- Continuation field
- Statement field

In ANSI format, the first five characters contain the line number and padding blanks. The sixth character is nonblank and nonzero if the line is a continuation of the last line. The VSI Fortran statement field begins at the seventh character and the line terminates with the 72nd character. Any characters after the 72nd character are ignored.

In tab format, the optional line number appears first on the line and is terminated by a tab character. If the character after the tab is a nonzero digit, that digit is the continuation field. The character after the continuation field begins the statement field. If the character after the tab is a nondigit character, that character begins the statement field.

B.1.1. Token and Placeholder Definitions

The bodies of VSI Fortran tokens and placeholders should be entered as legal source lines in tab format. This allows LSE to determine the fields and permit the lines to contain statement number and continuation fields. If ANSI_FORMAT is specified, LSE converts the body to ANSI format when the

body is expanded into the source file. A placeholder appearing in the statement number field is limited to five characters and must be a terminal placeholder.

B.1.2. Entering and Erasing Text

When a placeholder is erased from the statement number field in ANSI_FORMAT mode, it is replaced with blanks.

When a placeholder in the statement field is expanded, the statement number and continuation fields of the first line of the placeholder body are ignored and the statement field is inserted at the position vacated by the placeholder.

Note that the procedure for expanding tokens is identical.

B.1.3. Indentation

Indentation of FORTRAN statements is done only in the statement field,rather than at the beginning of a line as for other languages. Tab stops are set for the statement field only, with column 1 being the first character of the statement field.

An ENTER TAB or TAB command (bound to the TAB key) entered at the beginning of aline inserts a tab character in NOANSI mode. In ANSI mode, a tab at the beginning of a line moves the cursor to the statement field. Erasing one character backwards at that point puts the cursor in the continuation field.

An ENTER TAB or TAB command entered at the beginning of the statement field inserts the current indentation.

B.2. VSI COBOL

Within the VSI COBOL placeholders, you see the following three notations:

- [placeholder ..]
- [placeholder]...
- [placeholder]....

In the notation [placeholder ..], the space and two dots following the placeholder indicate that, upon expansion, you will see more details of the placeholder and not just the keywords that appear within the brackets.

The notation [placeholder]... is a list placeholder. The three dots indicate that the placeholder will be duplicated upon expansion.

The notation [placeholder].... is a list placeholder followed by punctuation, in this case, a period. The first three dots indicate that the placeholder will be duplicated upon expansion. Upon removal of the duplicated placeholder,a period will end the line.

Appendix C. Packages

LSE provides a mechanism for defining your own packages. The package facility includes two DECTPU sets of procedures to help you write your own packages.

C.1. DECTPU Procedures for the Package Facility

The DECTPU procedures generate the appropriate DEFINE TOKEN and DEFINEPLACEHOLDER commands for each routine or parameter. The DECTPU procedures,indicated by the /ROUTINE_EXPAND qualifier, generate a token definition for the routine name. The DECTPU procedures, indicated by the /PARAMETER_EXPAND qualifier, generate one or more placeholder definitions for each parameter name.

LSE comes with two sets of predefined DECTPU procedures that you can use to perform these expansions. For each language associated with the package, there must be a ROUTINE_EXPAND and PARAMETER_EXPAND procedure. You can specify these procedures with the /ROUTINE_EXPAND and /PARAMETER_EXPAND qualifiers. The value that you specify for these qualifiers in the DEFINEPACKAGE command must be a prefix shared by all the corresponding procedures. LSE determines the actual procedure name by concatenating the prefix value and the appropriate language name.

The procedures supplied with LSE are as follows:

- LSE$PKG_EXPAND_ROUT_ADA
- LSE$PKG_EXPAND_PARM_ADA
- LSE$PKG_EXPAND_ROUT_BASIC
- LSE$PKG_EXPAND_PARM_BASIC
- LSE$PKG_EXPAND_ROUT_BLISS
- LSE$PKG_EXPAND_PARM_BLISS
- LSE$PKG_EXPAND_ROUT_C
- LSE$PKG_EXPAND_PARM_C
- LSE$PKG_EXPAND_ROUT_COBOL
- LSE$PKG_EXPAND_PARM_COBOL
- LSE$PKG_EXPAND_ROUT_FORTRAN
- LSE$PKG_EXPAND_PARM_FORTRAN
- LSE$PKG_EXPAND_ROUT_PASCAL
- LSE$PKG_EXPAND_PARM_PASCAL
- LSE$PKG_EXPAND_ROUT_PLI
- LSE$PKG_EXPAND_PARM_PLI

You use these routines by specifying the following:

```
/ROUTINE_EXPAND = LSE$PKG_EXPAND_ROUT_
/PARAMETER_EXPAND = LSE$PKG_EXPAND_PARM_
```

If you want to write your own DECTPU procedures for these purposes, they must conform to the following restrictions:

- When LSE needs to generate a token definition from a routine definition, it calls the DECTPU procedure specified by the /ROUTINE_EXPAND qualifier. A typical DECTPU procedure for the/ ROUTINE_EXPAND qualifier appears as follows:

```
PROCEDURE my_routine_expand_somelanguage
LOCAL command_string, {other tpu local variables}...;
 . . .
command_string :=     'DEFINE TOKEN '
                    + routine_name
                    + '/LANGUAGE = somelanguage '
                    + <any other qualifiers for the DEFINE TOKEN
 command>;
LSE$DO_COMMAND (command_string) ;

< tpu code to generate the body of the token, using LSE$DO_COMMAND>

LSE$DO_COMMAND ('END DEFINE');

ENDPROCEDURE;
```

  All the information included in the DEFINE ROUTINE command is passed to your DECTPU routine by means of the DECTPU global variables. The following global variables are defined:

Variable	Description
LSE$PKG_ROUT_NAME	Contains the name of the routine being defined, enclosed in quotes (").
LSE$PKG_ROUT_LANG	Contains the name of the language.
LSE$PKG_ROUT_DESC	Contains the value of the /DESCRIPTION parameter.
LSE$PKG_ROUT_TOP	Contains the value of the /TOPIC parameter.
LSE$PKG_ROUT_PACK	Contains the package name.
LSE$PKG_ROUT_PARM	Contains the list of parameters associated with the routine being defined. Each parameter is enclosed in quotes and separated from the next by a carriage return, line feed pair; that is, the TPU string ASCII(13) + ASCII(10).
LSE$PKG_ROUT_OPT	Contains a list of flags, in one-to-one correspondence to the list of parameters. Each flag can be either O, indicating that the parameter is optional, or R, indicating that the parameter is required. Each flag is separated from the next by a single-space character.
LSE$PKG_ROUT_MECH	Contains a list of flags, in one-to-one correspondence to the list of parameters. Each flag can have one of the following values:

Variable	Description
	○ V—By value ○ R—By reference ○ D—By descriptor ○ U—Unknown Each flag is separated from the next by a single-space character.

- When LSE needs to generate a placeholder definition from a parameter definition,it calls the DECTPU procedure specified by the /PARAMETER_EXPAND qualifier of the package. A typical DECTPU procedure for /PARAMETER_EXPAND appears as follows:

```
PROCEDURE my_parameter_expand_somelanguage
LOCAL command_string, {other tpu local variables}...;
 . . .
command_string :=     'DEFINE PLACEHOLDER '
                    + routine_name
                    + '/LANGUAGE = somelanguage '
                    + '/TYPE = TERMINAL/SEPARATOR = ","'
                    + {other qualifiers for the DEFINE PLACEHOLDER
 command};
LSE$DO_COMMAND (command_string) ;
 . . .
ENDPROCEDURE;
```

The following global variables are defined for use by the DECTPU procedure specified by the /PARAMETER_EXPAND qualifier:

Variable	Description
LSE$PKG_PARA_NAME	Contains the name of the parameter to be defined.
LSE$PKG_PARA_LANG	Contains the name of the language.

C.2. Example Procedures

This section presents the TPU expansion procedures for Pascal and some of the support routines. The first TPU procedure, LSE$PKG_EXPAND_ROUT_PASCAL, defines a token for a package routine. It calls other TPU procedures that you can use as is or redefine according to your needs.

The second procedure,LSE$PKG_EXPAND_PARM_PASCAL, defines two placeholders for each parameter. Because the Pascal system-service routines are in a keyword format (for example, %[p1 :=%{p1}%]%), a placeholder must be defined for p1 and p1 := %{p1}%. The first placeholder is defined in the procedure LSE$PKG_DEFINE_PARAMETER, and the second in LSE $PKG_EXPAND_PARAM_PASCAL.

The following called procedures are also listed:

- LSE$PKG_PAD_NAME
- LSE$PKG_DEFINE_TOKEN

- LSE$PKG_GET_PARAM
- LSE$PKG_DEFINE_PARAMETER

Note that the TPU built-in procedure, change_case, is called to force the case of expansions. You can modify the expansion routines to use CHANGE_CASE to follow any case convention you want.

```
PROCEDURE lse$pkg_expand_rout_pascal
!++
! FUNCTIONAL DESCRIPTION:
!
!   This routine generates a Pascal token definition from a parameter
!   definition using keyword syntax.
!
! FORMAL PARAMETERS:
!
!   None
!
! IMPLICIT INPUTS:
!
!   LSE$PKG_ROUT_NAME
!       The name of the routine to be defined.
!
!   LSE$PKG_ROUT_PARM
!       The list of parameters of the routine separated by spaces.
!
!   LSE$PKG_ROUT_OPT
!       A list of flags in one-to-one correspondence with the list of
!       parameters.  Each flag can be either O, indicating optional, or
!       R, indicating required.  Each flag is separated from the next by a
!       space.
!
! IMPLICIT OUTPUTS:
!
!   None
!
! ROUTINE VALUE:
!
!   None
!
! SIDE EFFECTS:
!
!   A token definition is issued.
!-
    LOCAL
        proc_name,
        command_string,
        cur_param,
        cur_option,
        param_name,
        keyword_param,
        mech;

    ! Start the DEFINE TOKEN command.
    lse$pkg_define_token;

    ! Remove quotes from procedure name.
    proc_name := SUBSTR(LSE$PKG_ROUT_NAME, 2, LENGTH(LSE$PKG_ROUT_NAME)-2);
```

```
! Format the call with the procedure name in lowercase.
command_string := '"' + proc_name;
CHANGE_CASE(command_string,LOWER);

IF LSE$PKG_ROUT_PARM = '' THEN
    ! The call consists of just the procedure name
    command_string := command_string + '"';
    LSE$DO_COMMAND(command_string);

ELSE
    ! The call has parameters
    ! Form the first line of the call.
    !   First line is just the procedure name and open parenthesis.
    command_string := command_string + ' (';
    LSE$DO_COMMAND(command_string);
    ! Move a required parameter to the beginning of the list.
    !   This avoids a problem in erasing a comma after the first
    !   parameter if it is optional.
    lse$pkg_reorder_params (LSE$PKG_ROUT_PARM, LSE$PKG_ROUT_OPT);
    ! Loop for each parameter.
    LOOP
        EXITIF lse$pkg_get_param (cur_param,
                          cur_option,
                          LSE$PKG_ROUT_PARM,
                          LSE$PKG_ROUT_OPT) = 0;
        ! Remove passing mechanism .x suffix (x = v, d, or r).
        keyword_param := lse$pkg_remove_mech (cur_param, mech);
        ! Modify parameter names that conflict with Pascal keywords.
        IF keyword_param = "TYPE"
        THEN
            keyword_param := keyword_param + '_';
        ENDIF;
        lse$pkg_pad_name (keyword_param, param_name);
        ! Form the template line for the parameter.
        command_string := '"' + ASCII(9);
        IF cur_option = "O"
        THEN
            ! optional parameter
            command_string := command_string
                + '%[' + param_name + ' := %{' + cur_param + '}%]%'
        ELSE
            ! required parameter
            command_string := command_string
                + param_name + ' := %{' + cur_param + '}%'
            ENDIF;
        IF LSE$PKG_ROUT_PARM = '' THEN     ! No more parameters
            ! Complete the call statement.
            command_string := command_string + ')';
        ELSE
            ! Add a separator after the parameter.
            command_string := command_string + ',';
        ENDIF ;

        ! Make the line lowercase.
        CHANGE_CASE (command_string,LOWER);

        ! Add the line to the token definition.
        LSE$DO_COMMAND (command_string);
```

```
    ENDLOOP;
    ENDIF; ! parameter string is/isn't empty

    ! End the DEFINE TOKEN command
    LSE$DO_COMMAND ("end define") ;
ENDPROCEDURE

PROCEDURE lse$pkg_expand_parm_pascal
!++
! FUNCTIONAL DESCRIPTION:
!
!   This routine generates Pascal placeholder definitions from a parameter
!   definition for keyword syntax.
!
! FORMAL PARAMETERS:
!
!   None
!
! IMPLICIT INPUTS:
!
!   LSE$PKG_PARA_NAME
!       The name of the placeholder to define.
!
! IMPLICIT OUTPUTS:
!
!   None
!
! ROUTINE VALUE:
!
!   None
!
! SIDE EFFECTS:
!
!   Two placeholder definitions are issued.
!-
    LOCAL
        command_string,
        name_noquote,
        padded_key,
        keyword_name,
        mech;
    ! Define a placeholder for the parameter.
    lse$pkg_define_parameter('');
    ! Define a placeholder of the form "name := %{name}%".
    !
    This is done in case the parameter is optional.
    ! Strip the quotes off the name
    name_noquote := SUBSTR( LSE$PKG_PARA_NAME,
                            2,
                            LENGTH(LSE$PKG_PARA_NAME) - 2) ;
    ! Remove passing mechanism .x suffix (x = v, d, or r).
    keyword_name := lse$pkg_remove_mech (name_noquote, mech);
    ! Modify parameter names that conflict with Pascal keywords
    IF keyword_name = 'TYPE'
    THEN
        keyword_name := keyword_name + '_';
    ENDIF;
```

```
    lse$pkg_pad_name (keyword_name, padded_key) ;
    ! Do the DEFINE PLACEHOLDER command.
    command_string :=
        'define placeholder /language=pascal /separator="," "' +
                    padded_key + ' := %{' + name_noquote + '}%"' ;
    CHANGE_CASE (command_string, lower);
    LSE$DO_COMMAND (command_string) ;
    ! Do the body.
    command_string := '"' + padded_key + ' := %{' + name_noquote + '}%"' ;
    CHANGE_CASE (command_string, LOWER);
    LSE$DO_COMMAND (command_string) ;
    ! End the definition.
    LSE$DO_COMMAND ('end define') ;
ENDPROCEDURE

PROCEDURE lse$pkg_pad_name (cur_param, p_keyword)
!++
! FUNCTIONAL DESCRIPTION:
!
!   Pads a parameter name so that it is at least six characters long. This
!   is for use by keyword-style routine calls, so that the intermediate
!   assignment operations, which separate the keyword from the parameter
!   value, line up properly.
!
! FORMAL PARAMETERS:
!
!   cur_param
!       The parameter name to be padded.
!
!   p_keyword
!       The result of padding the parameter name.
!
! IMPLICIT INPUTS:
!
!   None
!
! IMPLICIT OUTPUTS:
!
!   None
!
! ROUTINE VALUE:
!
!   None
!
! SIDE EFFECTS:
!
!   p_keyword is set as indicated above
!-
    LOCAL
        len,
            !* the length of cur_param
        i ;
    p_keyword := cur_param;
    ! Pad the p_keyword so it's six letters long.
    !   This tends to make the keyword calls to system services look
    !   better.
    len := LENGTH (p_keyword) ;
    IF len < 6 THEN
```

```
        i := 0 ;
        LOOP EXITIF i = 6 - len ;
            p_keyword := p_keyword + " " ;
            i := i + 1 ;
        ENDLOOP ;
    ENDIF ;
ENDPROCEDURE

PROCEDURE lse$pkg_define_token
!++
! FUNCTIONAL DESCRIPTION:
!
!   This routine generates LSE DEFINE TOKEN commands for routines.
!   It issues only the DEFINE TOKEN token-name, with qualifiers, and
!   leaves the editor in a state ready to process the definition of
!   the body of the token.  This procedure is suitable for being
!   called from any procedure that needs to define a token from a
!   routine definition; the calling procedure is responsible for
!   defining the body of the routine and issuing the closing END
!   DEFINE command.
!
! FORMAL PARAMETERS:
!
!   None
!
! IMPLICIT INPUTS:
!
!   LSE$PKG_ROUT_NAME
!       The name of the routine to be defined.
!
!   LSE$PKG_ROUT_LANG
!       The name of the language for which to define the routine.
!
!   LSE$PKG_ROUT_DESC
!       The description string for the routine.
!
!   LSE$PKG_ROUT_TOP
!       The topic string for the routine.
!
! IMPLICIT OUTPUTS:
!
!   None
!
! ROUTINE VALUE:
!
!   None
!
! SIDE EFFECTS:
!
!   Begins a DEFINE TOKEN definition.  The next calls to
!   LSE$DO_COMMAND must complete the definition.
!
! MODIFICATION HISTORY:
!
!-
    LOCAL
        proc_name,
    ! name of routine being defined, with quotes removed
```

```
        command_string; ! command string to send to LSE$DO_COMMAND
    ! Form DEFINE TOKEN command string
    command_string :=
        'define token ' + LSE$PKG_ROUT_NAME
            + ' /language = ' + LSE$PKG_ROUT_LANG
            + ' /description = "' + LSE$PKG_ROUT_DESC
            + '" /topic_string = "' + LSE$PKG_ROUT_TOP + '"';
    ! Execute the DEFINE TOKEN command
    LSE$DO_COMMAND (command_string) ;
ENDPROCEDURE;

PROCEDURE lse$pkg_get_param (param, option, param_line, option_line)
!++
! FUNCTIONAL DESCRIPTION:
!
!   Return the first parameter and option from the given parameter
!   lists and option line, which removes them from the lists.
!
! FORMAL PARAMETERS:
!
!   param
!       On exit, this will be the first parameter from the param line.
!
!   option
!       On exit, this will be the first option field from the option line.
!
!   param_line
!       A list of parameters for a routine, as in LSE$PKG_ROUT_PARM.
!       On exit, the first parameter from the list will have been
!       removed.
!
!   option_line
!       A list of option flags for a routine's parameter list, as in
!       LSE$PKG_ROUT_OPT.  On exit, the first option from the list
!       will have been removed.
!
! IMPLICIT INPUTS:
!
!   None
!
! IMPLICIT OUTPUTS:
!
!   None
!
! ROUTINE VALUE:
!
!   0 - if there were no more parameters
!   1 - if a parameter name is returned
!
! SIDE EFFECTS:
!
!   param_line and option_line are changed as indicated above
!-
    LOCAL
        blank_idx ;
    ! ** location of blanks in parameter lines
    ! Locate a parameter in param_line.
    blank_idx := INDEX (param_line, ASCII(13)+ASCII(10) );
```

```
    ! Return if no more parameters.
    IF blank_idx <= 1 THEN
        param_line := '' ;
        RETURN (0) ;
    ENDIF ;
    ! Get parameter, stripping off the outside set of quotes.
    param := SUBSTR (param_line, 2, blank_idx - 3) ;
    ! Remove parameter from param_line.
    param_line := SUBSTR (param_line, blank_idx + 2, LENGTH(param_line) ) ;
    ! Get option and remove from option_line.
    option := SUBSTR (option_line, 1, 1) ;
    option_line := SUBSTR (option_line, 3, LENGTH(option_line) ) ;
    RETURN (1) ;
ENDPROCEDURE

PROCEDURE lse$pkg_define_parameter(qualifiers)
!++
! FUNCTIONAL DESCRIPTION:
!
!   This procedure issues a standard DEFINE PARAMETER command for the
! parameter currently being expanded. This routine is suitable for being
! called from any procedure that needs to define a placeholder from a
! parameter. Note that unlike lse$pkg_define_token, this routine generates
! a complete placeholder definition.
!
! FORMAL PARAMETERS:
!
!   qualifiers
!       A string containing any additional qualifiers to be added to the
!       placeholder definition.  Most commonly, this will be either empty
!       or just a separator definition (for example, '/SEPARATOR=","').
!       Note that the parameter must be a complete qualifier or sequence
!       of qualifier, in legal LSE syntax.  Furthermore, since this
!       routine automatically adds a /type=terminal and a
!       /language=lse$pkg_para_lang to the placeholder definitions,
!       these two qualifiers may NOT be included in the qualifiers
!       parameters.
!
!
! IMPLICIT INPUTS:
!
!   LSE$PKG_PARA_NAME
!       The name of the placeholder to define.
!
!   LSE$PKG_PARA_LANG
!       The language for which to define the placeholder.
!
! IMPLICIT OUTPUTS:
!
!   None
!
! ROUTINE VALUE:
!
!   None
!
! SIDE EFFECTS:
!
!   A new placeholder is defined.
```

```
!-
    LOCAL
        command_string,
        name_noquote,
        mech;
    ! Form DEFINE PLACEHOLDER command string
    command_string :=
        'define placeholder ' + LSE$PKG_PARA_NAME +
        ' /type=terminal /language=' + LSE$PKG_PARA_LANG
        + qualifiers;
    ! Force to lowercase
    CHANGE_CASE (command_string, LOWER);
    ! Execute the DEFINE PLACEHOLDER command
    LSE$DO_COMMAND (command_string);
    ! Strip the quotes off the name
    name_noquote := SUBSTR(LSE$PKG_PARA_NAME, 2,
    LENGTH(LSE$PKG_PARA_NAME) - 2);
    ! Remove passing mechanism .x suffix (x = v, d, or r).
    name_noquote := lse$pkg_remove_mech (name_noquote, mech);
    ! Do the body line.
    command_string := '"The actual data you want to pass to parameter ' +
                name_noquote + '."';
    LSE$DO_COMMAND (command_string);
    ! Do a body line for the passing mechanism.
    IF mech = 'V' THEN
        LSE$DO_COMMAND('"The parameter is passed by value."');
    ELSE IF mech = 'R' THEN
        LSE$DO_COMMAND('"The parameter is passed by reference."');
    ELSE IF mech = 'D' THEN
        LSE$DO_COMMAND('"The parameter is passed by descriptor."');
    ENDIF; ENDIF; ENDIF;
    ! End the DEFINE PLACEHOLDER command
    LSE$DO_COMMAND ("end define")
ENDPROCEDURE

PROCEDURE lse$pkg_remove_mech(param_name, mech_char)
!++
! FUNCTIONAL DESCRIPTION:
!
!   This procedure removes a suffix from a parameter name of the form
!   name.suffix.  The suffix must be either v, d, or r and
!   indicate that the parameter is passed by value, descriptor, or
!   reference, respectively.
!
! FORMAL PARAMETERS:
!
!   param_name
!       The name of the parameter.
!
!   mech_char
!       Set to the suffix character removed from param_name (uppercase).
!
! IMPLICIT INPUTS:
!
!   None
!
! IMPLICIT OUTPUTS:
!
```

```
!   None
!
! ROUTINE VALUE:
!
!   The parameter name without the .suffix.
!
! SIDE EFFECTS:
!
!   None
!-
    LOCAL
        param_length,
        mech_suffix,
        mech_separator;
    mech_char := '';
    param_length := LENGTH (param_name) ;
    IF param_length < 2 THEN RETURN (param_name) ENDIF;
    ! Get last character from param_name.
    mech_suffix := SUBSTR(param_name, param_length, 1);
    ! Get second-to-last character from param_name.
    mech_separator := SUBSTR(param_name, param_length - 1, 1);
    CHANGE_CASE (mech_suffix, UPPER);
    IF ((mech_suffix = 'V') OR (mech_suffix = 'D') OR (mech_suffix = 'R'))
            AND (mech_separator = '.') THEN
        mech_char := mech_suffix;
        RETURN ( SUBSTR (param_name, 1, param_length - 2) ) ;
    ENDIF;
    RETURN (param_name);
ENDPROCEDURE
```

Appendix D. LSE and EVE Commands

Table D.1, "Corresponding EVE and LSE Commands" lists the EVE commands with the corresponding LSE commands.

Table D.1. Corresponding EVE and LSE Commands

EVE Command	LSE Command
@	None
ATTACH	ATTACH
BOTTOM	GOTO BOTTOM
BUFFER	GOTO BUFFER
CAPITALIZE WORD	CAPITALIZE WORD
CENTER LINE	CENTER LINE
CHANGE DIRECTION	CHANGE DIRECTION
CHANGE MODE	CHANGE TEXT_ENTRY_MODE
DCL	DCL
DEFINE KEY	DEFINE KEY
DELETE BUFFER	DELETE BUFFER
DELETE WINDOW	DELETE WINDOW
DO	GOTO COMMAND
END OF LINE	GOTO LINE/BOUND/FORWARD
ENLARGE WINDOW	ENLARGE WINDOW
ERASE CHARACTER	ERASE/TO CHARACTER/REVERSE
ERASE LINE	ERASE/TO LINE/BEGINNING/FORWARD
ERASE PREVIOUS WORD	ERASE WORD/PREVIOUS
ERASE START OF LINE	ERASE/TO LINE/BEGINNING/REVERSE
ERASE WORD	ERASE WORD/NEXT
EXIT	EXIT
EXTEND ALL	EXTEND *
EXTEND EVE	EXTEND
EXTEND THIS	EXTEND /INDICATED
EXTEND TPU	DO /TPU
FILL	FILL
FILL PARAGRAPH	FILL
FILL RANGE	FILL
FIND	SEARCH
FORWARD	SET FORWARD
GET FILE	GOTO FILE

EVE Command	LSE Command
GOTO	GOTO MARK
HELP	HELP
INCLUDE FILE	READ
INSERT HERE	PASTE
INSERT MODE	SET INSERT
INSERT PAGE BREAK	None
LEARN	DEFINE KEY/LEARN
LINE	LINE
LOWERCASE WORD	LOWERCASE WORD
MARK	SET MARK
MOVE BY LINE	GOTO LINE/BREAK
MOVE BY PAGE	GOTO PAGE
MOVE BY WORD	GOTO WORD/BEGINNING/CURRENT
MOVE DOWN	GOTO CHARACTER/VERTICALLY/ FORWARD
MOVE LEFT	GOTO CHARACTER/HORIZONTALLY/ REVERSE
MOVE RIGHT	GOTO CHARACTER/HORIZONTALLY/ FORWARD
MOVE UP	GOTO CHARACTER/VERTICALLY/REVERSE
NEXT SCREEN	GOTO SCREEN/FORWARD
NEXT WINDOW	NEXT WINDOW
ONE WINDOW	ONE WINDOW
OTHER WINDOW	NEXT WINDOW
OVERSTRIKE MODE	SET OVERSTRIKE
PREVIOUS SCREEN	GOTO SCREEN/REVERSE
PREVIOUS WINDOW	PREVIOUS WINDOW
QUIT	QUIT
QUOTE	QUOTE
RECALL	RECALL
REFRESH	REFRESH
REMEMBER	END DEFINE
REMOVE	CUT
REPEAT	REPEAT
REPLACE	SUBSTITUTE
RESET	None
RESTORE	UNERASE
RESTORE CHARACTER	UNERASE CHARACTER
RESTORE LINE	UNERASE LINE

EVE Command	LSE Command
RESTORE SENTENCE	None
RESTORE WORD	UNERASE WORD
RETURN	ENTER LINE
REVERSE	SET REVERSE
SAVE EXTENDED EVE	SAVE SECTION
SAVE EXTENDED TPU	SAVE SECTION
SELECT	TOGGLE SELECT_MARK
SET CURSOR BOUND	SET CURSOR BOUND
SET CURSOR FREE	SET CURSOR FREE
SET FIND NOWHITESPACE	SET SEARCH NOSPAN_SPACE
SET FIND WHITESPACE	SET SEARCH SPAN_SPACE
SET LEFT MARGIN	SET LEFT MARGIN
SET NOPENDING DELETE	SET MODE NOPENDING_DELETE
SET NOWRAP	SET NOWRAP
SET PENDING DELETE	SET MODE PENDING_DELETE
SET RIGHT MARGIN	SET RIGHT MARGIN
SET SCROLL MARGINS	SET SCROLL_MARGINS
SET SHIFT KEY	DEFINE KEY /STATE=GOLD
SET TABS AT	None
SET TABS EVERY	None
SET TABS INVISIBLE	SET MODE TABS=INVISIBLE
SET TABS VISIBLE	SET MODE TABS=VISIBLE
SET WIDTH	SET SCREEN WIDTH
SET WILDCARD ULTRIX	SET SEARCH PATTERN=ULTRIX
SET WILDCARD VMS	SET SEARCH PATTERN=VMS
SET WRAP	SET WRAP
SHIFT LEFT	SHIFT/REVERSE
SHIFT RIGHT	SHIFT/FORWARD
SHOW	SHOW BUFFER
SHOW KEY	SHOW KEY
SHOW SUMMARY	SHOW SUMMARY
SHOW SYSTEM BUFFERS	SHOW BUFFERS/SYSTEM_BUFFERS
SHOW WILDCARD	None
SHRINK WINDOW	SHRINK WINDOW
SPACE	ENTER SPACE
SPAWN	SPAWN
SPELL	SPELL
SPLIT WINDOW	SPLIT WINDOW

EVE Command	LSE Command
START OF LINE	GOTO LINE/BOUND/REVERSE
STORE TEXT	None
TAB	TAB
TOP	GOTO TOP
TPU	DO/TPU "string "
TWO WINDOWS	TWO WINDOWS
UNDEFINE KEY	DELETE KEY
UPPERCASE WORD	UPPERCASE WORD
WHAT LINE	WHAT LINE
WILDCARD FIND	None
WRITE FILE	WRITE

Appendix E. Portable and VMSLSE Commands

This appendix contains a translation table that lists the Portable commands with their corresponding VMSLSE equivalents.

- There are cases in which many individual Portable commands are used to establish a set environment. Because the VMSLSE command language requires the setting of the complete environment in a single command line, there is not a simple VMSLSE equivalent for these Portable commands. Therefore, their listings typically show the corresponding VMSLSE base command, sometimes with one qualifier, and an ellipsis (. . .) to indicate that the VMSLSE command is not complete. For example:

SET LANGUAGE ANSI FORTRAN ON |OFF [defined_language] [1]	MODIFY LANGUAGE /FORTRAN=[NO]ANSI_FORMAT ...
SET LANGUAGE BRACKETED COMMENTS begin_string end_string [add_remove [defined_language]] [1]	MODIFY LANGUAGE /COMMENT= (BEGIN= string_list,END=string_list) ...

  Such commands are footnoted with [1] in *Table E.1, "Portable to VMSLSE Commands"*.

- The SET LANGUAGE commands are shown as being equivalent to the VMSLSE command MODIFY LANGUAGE.

  There are no equivalents to MODIFY LANGUAGE for the other complex DEFINE commands (PLACEHOLDER, TOKEN, etc.).

- Some commands have to be enabled using the portable command ENABLE VMS INTEGRATION. These commands are shown at the end of *Table E.1, "Portable to VMSLSE Commands"* in two separate table sections.

Note

Any Portable command can be issued as a VMSLSE command by prefixing the command with PLSE (such as PLSE SET PROMPT KEYPAD VMSLSE).

Table E.1. Portable to VMSLSE Commands

Portable Command	**VMSLSE Equivalent**
ALIGN [column]	ALIGN[/COMMENT_COLUMN=column]
ATTACH [process_name]	ATTACH [process_name]
BALANCE WINDOWS	No equivalent—See SET SCREEN BALANCE_WINDOWS
BOTTOM	GOTO BOTTOM
CAPITALIZE	CAPITALIZE WORD
CENTER LINE	CENTER LINE

Portable Command	**VMSLSE Equivalent**
CHANGE CASE	CHANGE CASE
CHECK LANGUAGE DEFINITIONS [defined_name]	CHECK LANGUAGE/DEFINITIONS defined_name
CHECK LANGUAGE HELP [defined_name]	CHECK LANGUAGE/HELP_INTERFACE defined_name
CLI cli_command	DCL cli_command
CLOSE	CLOSE BUFFER
CLOSE BUFFER [buffer_name]	No equivalent
CLOSE FILE file_spec	No equivalent
COLLAPSE [depth]	COLLAPSE[/DEPTH=depth]
COMPILE [compile_command]	COMPILE [compile_command]
COMPILE REVIEW [compile_command]	COMPILE/REVIEW [compile_command]
COPY [user_paste_buffer]	CUT/NOERASE[/BUFFER= user_paste_buffer |/CLIPBOARD]
COPY APPEND [user_paste_buffer]	CUT/NOERASE/APPEND[/BUFFER= user_paste_buffer |/CLIPBOARD]
CUT [user_paste_buffer]	CUT[/BUFFER=user_paste_buffer |/ CLIPBOARD]
CUT APPEND [buffer_name]	CUT/APPEND[/BUFFER=buffer_name |/ CLIPBOARD]
DELETE ADJUSTMENT [adjustment_name_wild [language_name_wild]]	DELETE ADJUSTMENT[/LANGUAGE= language_name_wild] adjustment_name_wild
DELETE ALIAS [alias_name_wild [language_name_wild]]	DELETE ALIAS[/LANGUAGE= language_name_wild] alias_name_wild
DELETE BUFFER [buffer_name]	DELETE BUFFER [buffer_name]
DELETE EXPAND	UNEXPAND
DELETE KEY user_key_name	DELETE KEY user_key_name
DELETE LANGUAGE language_name_wild	DELETE LANGUAGE language_name_wild
DELETE MARK [mark_name]	CANCEL MARK [mark_name]
DELETE MENU ENTRY menu_name menu_entry	No equivalent
DELETE MENU LABEL menu_label	No equivalent
DELETE MENU SEPARATOR menu_name number	No equivalent
DELETE PACKAGE package_name_wild	DELETE PACKAGE package_name_wild
DELETE PLACEHOLDER	DELETE PLACEHOLDER[/LANGUAGE=

Portable Command	**VMSLSE Equivalent**
[placeholder_name_wild [language_name_wild]]	language_name_wild] placeholder_name_wild
DELETE ROUTINE [routine_name_wild [package_name_wild]]	DELETE ROUTINE[/PACKAGE= package_name_wild] routine_name_wild
DELETE SELECTION MARK	CANCEL SELECT_MARK
DELETE TAB	UNTAB
DELETE TOKEN [token_name_wild [language_name_wild]]	DELETE TOKEN[/LANGUAGE= language_name_wild] token_name_wild
DELETE WINDOW	DELETE WINDOW
DISABLE GRAMMAR PREFIX prefix	No equivalent
DISABLE VMS INTEGRATION COMMANDS	No equivalent
ENABLE GRAMMAR PREFIX prefix help_library	No equivalent
ENABLE VMS INTEGRATION COMMANDS	No equivalent
END OF LINE	GOTO LINE/FORWARD/BOUND
ENLARGE WINDOW [number]	ENLARGE WINDOW number
ENTER COMMENT [BLOCK |LINE]	ENTER COMMENT[/BLOCK |/LINE]
ENTER LINE	ENTER LINE
ENTER PSEUDOCODE	ENTER PSEUDOCODE
ENTER SPACE	ENTER SPACE
ENTER SPECIAL ascii_code	ENTER SPECIAL ascii_code
ENTER TAB	ENTER TAB
ENTER TEXT text_string	ENTER TEXT text_string
ERASE CHARACTER	ERASE CHARACTER
ERASE COMMENT	No equivalent
ERASE END OF LINE	ERASE LINE/END/FORWARD
ERASE END OF WORD	ERASE WORD/FORWARD
ERASE LINE	ERASE LINE
ERASE NEXT CHARACTER	ERASE CHARACTER/FORWARD
ERASE NEXT LINE	ERASE LINE/BEGINNING/FORWARD
ERASE NEXT PLACEHOLDER ON |OFF	ERASE PLACEHOLDER/FORWARD/ [NO]GOTO_PLACEHOLDER
ERASE NEXT WORD	ERASE WORD/NEXT
ERASE NUM CHARS number	No equivalent
ERASE PLACEHOLDER ON |OFF	ERASE PLACEHOLDER/CURRENT/ [NO]GOTO_PLACEHOLDER
ERASE PREVIOUS CHARACTER	ERASE CHARACTER/REVERSE

Portable Command	**VMSLSE Equivalent**
ERASE PREVIOUS LINE	ERASE LINE/END/REVERSE
ERASE PREVIOUS PLACEHOLDER ON |OFF	ERASE PLACEHOLDER/REVERSE/ [NO]GOTO_PLACEHOLDER
ERASE PREVIOUS WORD	ERASE WORD/PREVIOUS
ERASE SELECTION	ERASE SELECTION
ERASE START OF LINE	ERASE LINE/BEGINNING/REVERSE
ERASE START OF WORD	ERASE WORD/REVERSE
ERASE WORD	ERASE WORD
EXACT SUBSTITUTE search_string replace_string ALL |SINGLE	SUBSTITUTE/CASE_MATCHING[/ALL] search_string replace_string
EXECUTE BUFFER LSE [buffer_name]	DO[/BUFFER=buffer_name]
EXECUTE BUFFER PLSE [buffer_name]	DO[/BUFFER=buffer_name]
EXECUTE BUFFER TPU [buffer_name]	DO/TPU[/BUFFER=buffer_name]
EXIT	EXIT
EXPAND [depth]	EXPAND[/DEPTH=depth]
EXTEND [procedure_name]	EXTEND [procedure_name]
EXTRACT ADJUSTMENT [adjustment_name_wild [language_name_wild]]	EXTRACT ADJUSTMENT[/LANGUAGE= language_name_wild] adjustment_name_wild
EXTRACT ALIAS [alias_name_wild [language_name_wild]]	EXTRACT ALIAS[/LANGUAGE= language_name_wild] alias_name_wild
EXTRACT LANGUAGE language_name_wild	EXTRACT LANGUAGE language_name_wild
EXTRACT NEW ADJUSTMENT [adjustment_name_wild [language_name_wild]]	EXTRACT ADJUSTMENT/NEW [/LANGUAGE=language_name_wild] adjustment_name_wild
EXTRACT NEW ALIAS [alias_name_wild [language_name_wild]]	EXTRACT ALIAS/NEW [/LANGUAGE=language_name_wild] alias_name_wild
EXTRACT NEW LANGUAGE language_name_wild	EXTRACT LANGUAGE/NEW language_name_wild
EXTRACT NEW PACKAGE package_name_wild	EXTRACT PACKAGE/NEW package_name_wild
EXTRACT NEW PLACEHOLDER [placeholder_name_wild [language_name_wild]]	EXTRACT PLACEHOLDER/NEW

Portable Command	**VMSLSE Equivalent**
	[/LANGUAGE=language_name_wild] placeholder_name_wild
EXTRACT NEW ROUTINE [routine_name_wild [package_name_wild]]	EXTRACT ROUTINE/NEW[/LANGUAGE= language_name_wild] routine_name_wild
EXTRACT NEW TOKEN [token_name_wild [language_name_wild]]	EXTRACT TOKEN/NEW[/LANGUAGE= language_name_wild] token_name_wild
EXTRACT PACKAGE package_name_wild	EXTRACT PACKAGE package_name_wild
EXTRACT PLACEHOLDER [placeholder_name_wild [language_name_wild]]	EXTRACT PLACEHOLDER[/LANGUAGE= language_name_wild] placeholder_name_wild
EXTRACT ROUTINE [routine_name_wild [package_name_wild]]	EXTRACT ROUTINE[/LANGUAGE= language_name_wild] routine_name_wild
EXTRACT TOKEN [token_name_wild [language_name_wild]]	EXTRACT TOKEN[/LANGUAGE= language_name_wild] token_name_wild
FETCH [element_name [element_id [remark]]]	[SET CMS/REMARK=remark] CMS FETCH[/GENERATION=element_id] [element_name] [remark]
FILL [column]	FILL[/COMMENT_COLUMN=column]
FIND OCCURRENCES	FIND/INDICATED
FOCUS	FOCUS
GOTO BUFFER buffer_name	GOTO BUFFER buffer_name
GOTO COMMAND [lse_command]	GOTO COMMAND
GOTO DECLARATION	GOTO DECLARATION/INDICATED
GOTO MARK mark_name	GOTO MARK mark_name
GOTO REVIEW	GOTO REVIEW
GOTO SOURCE	GOTO SOURCE
HELP [help_topic_wild]	HELP [help_topic_wild]
HELP INDICATED	HELP/INDICATED
HELP KEY user_key_name	No equivalent
HELP KEYPAD	HELP/KEYPAD
INCLUDE FILE file_spec	INCLUDE file_spec
INDENT LEFT	CHANGE INDENTATION/REVERSE
INDENT RIGHT	CHANGE INDENTATION/FORWARD
LINE number [procedure_name]	LINE number [procedure_name]
LOWERCASE	LOWERCASE WORD
LSE lse_command	LSE lse_command
MOVE DOWN	GOTO CHARACTER/FORWARD/VERTICAL

Portable Command	VMSLSE Equivalent
MOVE UP	GOTO CHARACTER/REVERSE/VERTICAL
NEW ADJUSTMENT adjustment_name [defined_language] [1]	DEFINE ADJUSTMENT ...
NEW ALIAS alias_name [defined_language] [1]	DEFINE ALIAS ...
NEW BUFFER new_buffer_name	GOTO BUFFER/CREATE new_buffer_name
NEW FILE file_spec	GOTO FILE/CREATE file_spec
NEW KEY user_key_name lse_command help_topic remark legend tpu_command [1]	DEFINE KEY ...
NEW LANGUAGE language_name [1]	DEFINE LANGUAGE ...
NEW LEARN KEY user_key_name [1]	DEFINE KEY/LEARN ...
NEW MARK mark_name	SET MARK mark_name
NEW MENU ENTRY menu_name menu_label before_menu_entry	No equivalent
NEW MENU LABEL menu_label tpu_command mnemonic_character	No equivalent
NEW MENU SEPARATOR menu_name before_menu_entry	No equivalent
NEW PACKAGE package_name [1]	DEFINE PACKAGE ...
NEW PLACEHOLDER placeholder_name [1]	DEFINE PLACEHOLDER ...
NEW ROUTINE routine_name defined_package [1]	DEFINE ROUTINE ...
NEW SELECTION MARK	SET SELECT_MARK
NEW TOKEN token_name token_type [defined_language] [1]	DEFINE TOKEN ...
NEW WINDOW [number]	SPLIT WINDOW [number]
NEXT BUFFER	NEXT BUFFER
NEXT CHARACTER	GOTO CHARACTER/FORWARD
NEXT END OF LINE	GOTO LINE/FORWARD/BREAK
NEXT ERROR	NEXT ERROR
NEXT PAGE	GOTO PAGE/FORWARD
NEXT PLACEHOLDER	GOTO PLACEHOLDER/FORWARD
NEXT SCREEN	GOTO SCREEN/FORWARD
NEXT START OF LINE	GOTO LINE/FORWARD
NEXT WINDOW	NEXT WINDOW
NEXT WORD	GOTO WORD/FORWARD
ONE WINDOW	ONE WINDOW
OPEN FILE file_spec	GOTO FILE file_spec
OPEN SELECTED FILE	No equivalent

Portable Command	**VMSLSE Equivalent**
OVERVIEW SOURCE	VIEW SOURCE/DEPTH=1
PASTE [user_paste_buffer]	PASTE[/BUFFER[=user_paste_buffer] |/ CLIPBOARD]
PATTERN EXACT SUBSTITUTE pattern_search_string replace_string ALL |SINGLE	SUBSTITUTE/PATTERN/CASE_MATCHING [/ALL] pattern_search_string replace_string
PATTERN SEARCH pattern_search_string	SEARCH/PATTERN pattern_search_string
PATTERN SUBSTITUTE pattern_search_string replace_string ALL |SINGLE	SUBSTITUTE/PATTERN[/ALL] pattern_search_string replace_string
PLSE lse_command	PLSE lse_command
POSITION CURSOR line column	No equivalent
PREVIOUS BUFFER	PREVIOUS BUFFER
PREVIOUS CHARACTER	GOTO CHARACTER/REVERSE
PREVIOUS END OF LINE	GOTO LINE/REVERSE/END
PREVIOUS ERROR	PREVIOUS ERROR
PREVIOUS PAGE	GOTO PAGE/REVERSE
PREVIOUS PLACEHOLDER	GOTO PLACEHOLFER/REVERSE
PREVIOUS SCREEN	GOTO SCREEN/REVERSE
PREVIOUS START OF LINE	GOTO LINE/PREVIOUS
PREVIOUS WINDOW	PREVIOUS WINDOW
PREVIOUS WORD	GOTO WORD/PREVIOUS
QUIT	QUIT
QUOTE	QUOTE
QUOTE KEYNAME	No equivalent
RECOVER BUFFER file_spec	RECOVER BUFFER file_spec
REDO	REDO
REFRESH	REFRESH
REPEAT number lse_command	REPEAT number lse_command
REPLACE [element_name [element_id [remark]]]	[SET CMS/REMARK=remark] REPLACE[/GENERATION=element_id] [element_name]
RESERVE [element_name [element_id [remark]]]	[SET CMS/REMARK=remark] RESERVE[/GENERATION=element_id] [element_name]
RESTORE [restore_option]	UNERASE [restore_option]
REVIEW	REVIEW
REVIEW BUFFER buffer_name	REVIEW buffer_name

Portable Command	**VMSLSE Equivalent**
REVIEW FILE file_spec	REVIEW/FILE=file_spec
SAVE AS file_spec	WRITE file_spec
SAVE ENVIRONMENT file_spec	SAVE ENVIRONMENT file_spec
SAVE ENVIRONMENT CHANGES file_spec	SAVE ENVIRONMENT/NEW file_spec
SAVE FILE file_spec	WRITE file_spec
SAVE SECTION file_spec	SAVE SECTION file_spec
SAVE SELECTION file_spec	WRITE/SELECT_REGION file_spec
SAVE VISIBLE file_spec	WRITE/VISIBLE file_spec
SEARCH search_string	SEARCH search_string
SELECT	TOGGLE SELECT MARK
SELECT ALL	SELECT ALL
SET ADJUSTMENT COMPRESS ON |OFF [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/[NO]COMPRESS ...
SET ADJUSTMENT COUNT ON |OFF [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/[NO]COUNT ...
SET ADJUSTMENT CURRENT current_indentation [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/CURRENT= current_indentation ...
SET ADJUSTMENT INHERIT inherit_keyword [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/[NO]INHERIT= inherit_keyword ...
SET ADJUSTMENT OVERVIEW ON |OFF [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/[NO]OVERVIEW ...
SET ADJUSTMENT PATTERN adjustment_pattern [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT ...
SET ADJUSTMENT PREFIX ADJUSTMENT adjustment_value [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/PREFIX= ...
SET ADJUSTMENT PREFIX INDENTATION indentation_value [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/PREFIX= ...
SET ADJUSTMENT SUBSEQUENT subsequent_indentation [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/SUBSEQUENT= subsequent_indentation ...
SET ADJUSTMENT UNIT ON |OFF [defined_adjustment [defined_language]] [1]	DEFINE ADJUSTMENT/[NO]UNIT ...

Portable Command	**VMSLSE Equivalent**
SET ALIAS EXPAND TEXT text_string [defined_alias [defined_language]]	DEFINE ALIAS ...
SET BALANCE WINDOWS ON |OFF	SET SCREEN [NO]BALANCE_WINDOW
SET BELL ALL OFF	SET MODE BELL=NONE
SET BELL ALL ON	SET MODE BELL=ALL
SET BELL BROADCAST ON |OFF	SET MODE BELL=[NO]BROADCAST
SET BUFFER AUTO ERASE ON |OFF	SET [NO]AUTO_ERASE
SET BUFFER CLOSE READ_ONLY	SET READ_ONLY
SET BUFFER CLOSE SAVE	SET WRITE
SET BUFFER DIRECTION FORWARD	SET FORWARD
SET BUFFER DIRECTION REVERSE	SET REVERSE
SET BUFFER INDENTATION level	SET INDENTATION level
SET BUFFER JOURNALING ON |OFF	SET [NO]JOURNALING
SET BUFFER LANGUAGE language_name	SET LANGUAGE language_name
SET BUFFER LEFT MARGIN column	SET LEFT_MARGIN column
SET BUFFER MODIFIABLE ON |OFF	SET [NO]MODIFY
SET BUFFER OUTPUT FILE file_spec	SET OUTPUT_FILE file_spec
SET BUFFER OVERVIEW ON |OFF	SET [NO]OVERVIEW
SET BUFFER RIGHT MARGIN column	SET RIGHT_MARGIN column
SET BUFFER TAB INCREMENT number	SET TAB_INCREMENT number
SET BUFFER TEXT INSERT	SET INSERT
SET BUFFER TEXT OVERSTRIKE	SET OVERSTRIKE
SET BUFFER WRAP ON |OFF	SET [NO]WRAP
SET CLIPBOARD ON |OFF	COPY[/CLIPBOARD] ... | CUT[/CLIPBOARD] ... | PASTE[/CLIPBOARD] ...
SET COMMAND LANGUAGE command_language	SET COMMAND LANGUAGE command_language
SET CURSOR cursor_option	SET CURSOR cursor_option
SET DIRECTORY DEFAULT directory_spec	SET DEFAULT_DIRECTORY directory_spec
SET DIRECTORY READONLY directory_spec ADD |REMOVE	SET DIRECTORY[/READ_ONLY |/WRITE] directory_spec
SET DIRECTORY SOURCE directory_spec	SET SOURCE_DIRECTORY directory_spec
SET FONT font_attribute	SET FONT font_attribute
SET HEIGHT number	SET SCREEN HEIGHT=number
SET KEYPAD keypad_name	SET MODE KEYPAD=keypad_name
SET LANGUAGE ANSI FORTRAN ON |OFF [defined_language] [1]	MODIFY LANGUAGE/FORTRAN= [NO]ANSI_FORMAT ...

Portable Command	VMSLSE Equivalent
SET LANGUAGE BRACKETED COMMENTS begin_string end_string [add_remove [defined_language]] [1]	MODIFY LANGUAGE/COMMENT= (BEGIN=string_list,END=string_list) ...
SET LANGUAGE COMMENT ASSOCIATION comment_association [defined_language] [1]	MODIFY LANGUAGE/COMMENT= ASSOCIATED=comment_association ...
SET LANGUAGE COMPILE COMMAND compile_command [defined_language] [1]	MODIFY LANGUAGE /COMPILE_COMMAND= compile_command ...
SET LANGUAGE DIAGNOSTICS ON |OFF [defined_language] [1]	MODIFY LANGUAGE/CAPABILITIES= [NO]DIAGNOSTICS ...
SET LANGUAGE ESCAPES character_string [defined_language] [1]	MODIFY LANGUAGE/QUOTED_ITEM= ESCAPES=character_string ...
SET LANGUAGE EXPAND CASE case_type [defined_language] [1]	MODIFY LANGUAGE/EXPAND_CASE= case_type ...
SET LANGUAGE FILE TYPES text_string [add_remove [defined_language]] [1]	MODIFY LANGUAGE/FILE_TYPES= string_list ...
SET LANGUAGE FIXED COMMENTS text_string column [add_remove [defined_language]] [1]	MODIFY LANGUAGE/COMMENT= FIXED=(string,column) ...
SET LANGUAGE FORTRAN boolean [defined_language] [1]	No equivalent
SET LANGUAGE HELP LIBRARY file_spec [defined_language] [1]	MODIFY LANGUAGE/HELP_LIBRARY= file_spec ...
SET LANGUAGE HELP TOPIC text_string [defined_language] [1]	MODIFY LANGUAGE/TOPIC_STRING= text_string ...
SET LANGUAGE IDENTIFIER CHARACTERS identifier_characters [defined_language] [1]	MODIFY LANGUAGE /IDENTIFIER_CHARACTERS= identifier_characters ...
SET LANGUAGE INITIAL STRING text_string [defined_language] [1]	MODIFY LANGUAGE/INITIAL_STRING= text_string ...
SET LANGUAGE LEFT MARGIN column [defined_language] [1]	MODIFY LANGUAGE/LEFT_MARGIN= column ...
SET LANGUAGE LINE COMMENTS text_string [add_remove [defined_language]] [1]	MODIFY LANGUAGE/COMMENT=LINE=

Portable Command	**VMSLSE Equivalent**
	string_list ...
SET LANGUAGE OPTIONAL DELIMIT begin_string end_string [defined_language] [1]	MODIFY LANGUAGE /PLACEHOLDER_DELIMITERS= (OPTIONAL=begin_string,end_string) ...
SET LANGUAGE OPTIONAL LIST DELIMIT begin_string end_string [defined_language] [1]	MODIFY LANGUAGE /PLACEHOLDER_DELIMITERS= (OPTIONAL_LIST=begin_string,end_string) ...
SET LANGUAGE OVERVIEW MINIMUM LINES number defined_language [1]	MODIFY LANGUAGE /OVERVIEW_OPTIONS=MINIMUM_LINES= number ...
SET LANGUAGE OVERVIEW TAB RANGE min_value max_value [defined_language] [1]	MODIFY LANGUAGE /OVERVIEW_OPTIONS=TAB_RANGE= (min_value,max_value) ...
SET LANGUAGE PSEUDOCODE DELIMIT begin_string end_string [defined_language] [1]	MODIFY LANGUAGE /PLACEHOLDER_DELIMITERS= PSEUDOCODE=(begin_string,end_string) ...
SET LANGUAGE PUNCTUATION CHARACTERS character_string [defined_language] [1]	MODIFY LANGUAGE /PUNCTUATION_CHARACTERS= character_string ...
SET LANGUAGE QUOTES character_string [defined_language] [1]	MODIFY LANGUAGE/QUOTED_ITEM= QUOTES=character_string ...
SET LANGUAGE REQUIRED DELIMIT begin_string end_string [defined_language] [1]	MODIFY LANGUAGE /PLACEHOLDER_DELIMITERS= REQUIRED=(begin_string,end_string) ...
SET LANGUAGE REQUIRED LIST DELIMIT begin_string end_string [defined_language] [1]	MODIFY LANGUAGE /REQUIRED_LIST= (begin_string,end_string) ...
SET LANGUAGE RIGHT MARGIN column [defined_language] [1]	MODIFY LANGUAGE/RIGHT_MARGIN= column ...
SET LANGUAGE TAB INCREMENT number [defined_language] [1]	MODIFY LANGUAGE/TAB_INCREMENT= number ...

Portable Command	VMSLSE Equivalent
SET LANGUAGE TAG TERMINATORS character_string [add_remove [defined_language]] [1]	MODIFY LANGUAGE/TAG_TERMINATORS= string_list ...
SET LANGUAGE TRAILING COMMENTS text_string [add_remove [defined_language]] [1]	MODIFY LANGUAGE/COMMENT= TRAILING=string_list ...
SET LANGUAGE VERSION text_string [defined_language] [1]	MODIFY LANGUAGE/VERSION=text_string ...
SET LANGUAGE WRAP boolean [defined_language] [1]	MODIFY LANGUAGE/WRAP boolean ...
SET MAX UNDO number	SET MAX_UNDO=number
SET MAXIMUM WINDOWS number	SET SCREEN MAXIMUM_WINDOW_NUMBER=number
SET MENU LABEL menu_label tpu_command mnemonic_character	No equivalent
SET MENU MNEMONICS boolean	No equivalent
SET MINIMUM WINDOW LENGTH=number	SET SCREEN MINIMUM_WINDOW_LENGTH number
SET NUMBER OF WINDOWS number	SET SCREEN WINDOW=number
SET PACKAGE HELP LIBRARY file_spec [defined_package] [1]	DEFINE PACKAGE/HELP_LIBRARY= file_spec ...
SET PACKAGE HELP TOPIC text_string [defined_package] [1]	DEFINE PACKAGE/TOPIC_STRING= text_string ...
SET PACKAGE LANGUAGE defined_language [add_remove [defined_package]] [1]	DEFINE PACKAGE/LANGUAGE= language_list ...
SET PACKAGE PARAMETER EXPAND text_string defined_package [1]	DEFINE PACKAGE /PARAMETER_EXPAND=text_string ...
SET PACKAGE ROUTINE EXPAND text_string defined_package [1]	DEFINE PACKAGE/ROUTINE_EXPAND= text_string ...
SET PENDING DELETE ON |OFF	SET MODE [NO]PENDING_DELETE
SET PLACEHOLDER AUTO SUBSTITUTE ON |OFF [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER /[NO]AUTO_SUBSTITUTE ...
SET PLACEHOLDER BODY LINE	DEFINE PLACEHOLDER/TYPE=

Portable Command	**VMSLSE Equivalent**
body_string indent_type indent_column- tab_or_space same_next_line [add_remove [defined_placeholder- [defined_language]]] [1]	NONTERMINAL ...
SET PLACEHOLDER DESCRIPTION description [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER/DESCRIPTION= description ...
SET PLACEHOLDER DUPLICATION duplication [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER/DUPLICATION= duplication ...
SET PLACEHOLDER HELP TOPIC help_topic [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER /TOPIC_STRING=help_topic ...
SET PLACEHOLDER INHERIT placeholder_name [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER /PLACEHOLDER=placeholder_name ...
SET PLACEHOLDER LEADING text_string [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER/LEADING= text_string ...
SET PLACEHOLDER MENU LINE body_string description menu_line_type- list_boolean [add_remove [defined_placeholder [defined_language]]] [1]	DEFINE PLACEHOLDER/TYPE=MENU ...
SET PLACEHOLDER PSEUDOCODE ON |OFF [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER /[NO]PSEUDOCODE [defined_placeholder] ...
SET PLACEHOLDER SEPARATOR text_string [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER/SEPARATOR= text_string ...
SET PLACEHOLDER TERMINAL LINE body_string [add_remove [defined_placeholder [defined_language]]] [1]	DEFINE PLACEHOLDER/TYPE= TERMINAL ...
SET PLACEHOLDER TRAILING text_string [defined_placeholder [defined_language]] [1]	DEFINE PLACEHOLDER/TRAILING= text_string ...

Portable Command	VMSLSE Equivalent
SET PRIMARY SELECTION MODEL selection_model	No equivalent
SET PROMPT ABORT user_key_name add_remove	No equivalent
SET PROMPT ALTERMINATOR user_key_name add_remove	No equivalent
SET PROMPT DIALOG CCT | DEFAULT	No equivalent
SET PROMPT DIALOG WINDOW	DEFINE KEY/DIALOG ... | SEARCH/DIALOG ... | WRITE/DIALOG ...
SET PROMPT EXPANDMENU prompt_keypad	No equivalent
SET PROMPT KEYPAD prompt_keypad	No equivalent
SET PROMPT TERMINATOR user_key_name add_remove	No equivalent
SET ROUTINE DESCRIPTION description [defined_routine [defined_package]] [1]	DEFINE ROUTINE/DESCRIPTION= description ...
SET ROUTINE HELP TOPIC help_topic [defined_routine [defined_package]] [1]	DEFINE ROUTINE/TOPIC_STRING= help_topic ...
SET ROUTINE PARAMETER parameter_name optional_or_required mechanism- [add_remove [defined_routine [defined_package]]] [1]	DEFINE ROUTINE ...
SET SAVE RELATED BUFFERS boolean	No equivalent
SET SCROLL MARGINS top_margin bottom_margin	SET SCROLL_MARGINS top_margin bottom_margin
SET SEARCH AUTO REVERSE OFF	SET SEARCH NOAUTO_REVERSE
SET SEARCH AUTO REVERSE ON	SET SEARCH AUTO_REVERSE
SET SEARCH CASE SENSITIVE OFF	SET SEARCH NOCASE_SENSITIVE
SET SEARCH CASE SENSITIVE ON	SET SEARCH CASE_SENSITIVE
SET SEARCH DIACRITICAL OFF	SET SEARCH NODIACRITICAL_SENSITIVE
SET SEARCH DIACRITICAL ON	SET SEARCH DIACRITICAL_SENSITIVE
SET SEARCH PATTERN search_pattern_name	SET SEARCH PATTERN search_pattern_name
SET SEARCH SPAN SPACE OFF	SET SEARCH NOSPAN_SPACE
SET SEARCH SPAN SPACE ON	SET SEARCH SPAN_SPACE
SET TABS HARD OFF	SET MODE TABS=NOHARD

Portable Command	**VMSLSE Equivalent**
SET TABS HARD ON	SET MODE TABS=HARD
SET TABS VISIBLE OFF	SET MODE TABS=NOVISIBLE
SET TABS VISIBLE ON	SET MODE TABS=VISIBLE
SET TOKEN BODY LINE body_string indent_type indent_column tab_or_space- same_next_line [add_remove [defined_token [defined_language]]] [1]	DEFINE TOKEN ...
SET TOKEN DESCRIPTION description [defined_token [defined_language]] [1]	DEFINE TOKEN/DESCRIPTION= description ...
SET TOKEN HELP TOPIC help_topic [defined_token [defined_language]] [1]	DEFINE TOKEN/TOPIC_STRING= help_topic ...
SET TOKEN INHERIT placeholder_name [defined_token [defined_language]] [1]	DEFINE TOKEN/PLACEHOLDER= placeholder_name ...
SET UNDO OFF	SET MODE UNDO=OFF
SET UNDO ON	SET MODE UNDO=ON
SET WIDTH number	SET SCREEN WIDTH=number
SHIFT LEFT	SHIFT/REVERSE
SHIFT LEFT number	REPEAT number SHIFT/REVERSE
SHIFT_RIGHT	SHIFT/FORWARD
SHIFT_RIGHT number	REPEAT number SHIFT/FORWARD
SHOW ADJUSTMENT [adjustment_name_wild [defined_language]]	SHOW ADJUSTMENT[/LANGUAGE= defined_language] [adjustment_name_wild]
SHOW ALIAS [alias_name_wild [defined_language]]	SHOW ALIAS[/LANGUAGE= defined_language] [alias_name_wild]
SHOW ATTRIBUTES	SHOW DEFAULT_DIRECTORY *or* SHOW DIRECTORY *or* SHOW MODE *or* SHOW SOURCE_DIRECTORY
SHOW BUFFER buffer_name_wild	SHOW BUFFER buffer_name_wild
SHOW KEY user_key_name_wild	SHOW KEY user_key_name_wild
SHOW LANGUAGE language_name_wild	SHOW LANGUAGE language_name_wild
SHOW LANGUAGE ROUTINE [routine_name_wild [defined_language]]	SHOW ROUTINE[/LANGUAGE= defined_language] routine_name_wild
SHOW MARK [mark_name_wild]	SHOW MARK [mark_name_wild]

Portable Command	**VMSLSE Equivalent**
SHOW MAX UNDO	SHOW MAX_UNDO
SHOW MAX UNDO buffer_name	No equivalent
SHOW PACKAGE [package_name_wild]	SHOW PACKAGE package_name_wild
SHOW PACKAGE ROUTINE [routine_name_wild [defined_package]]	SHOW ROUTINE[/PACKAGE= defined_package] [routine_name_wild]
SHOW PLACEHOLDER [placeholder_name_wild [defined_language]]	SHOW PLACEHOLDER[/LANGUAGE= defined_language] [placeholder_name_wild]
SHOW PROMPT ATTRIBUTES	No equivalent
SHOW SEARCH ATTRIBUTES	SHOW SEARCH
SHOW SUMMARY	SHOW SUMMARY
SHOW SYSTEM BUFFER [buffer_name_wild]	SHOW BUFFER/SYSTEM_BUFFERS [buffer_name_wild]
SHOW TOKEN [token_name_wild [defined_language]]	SHOW TOKEN[/LANGUAGE= defined_language] token_name_wild
SHOW VERSION	SHOW VERSION
SHOW WINDOW ATTRIBUTES	SHOW SCREEN
SHRINK WINDOW [number]	SHRINK WINDOW [number]
SPAWN [cli_command]	SPAWN [cli_command]
SPELL	SPELL
START OF LINE	GOTO LINE/REVERSE/BOUND
SUBSTITUTE search_string replace_string ALL | SINGLE	SUBSTITUTE[/ALL] search_string replace_string
TOGGLE INSERT OVERSTRIKE	CHANGE TEXT_ENTRY_MODE
TOP	GOTO TOP
TPU tpu_command	TPU tpu_command
TWO WINDOWS	TWO WINDOWS
UNDO	UNDO
UNRESERVE [element_name [element_id [remark]]]	[SET CMS/REMARK=remark] UNRESERVE[/GENERATION=element_id] [element_name]
UPPERCASE	UPPERCASE WORD
VIEW DEBUGGING SOURCE	VIEW SOURCE/DEBUG
VIEW_FILE file_spec	GOTO FILE/READ_ONLY file_spec
VIEW SOURCE depth	VIEW SOURCE/DEPTH=depth
WHAT LINE	WHAT LINE
Additional Commands if VMSSCA_ Prefix is Enabled	

Portable Command	**VMSLSE Equivalent**
ANALYZE [file_spec]	ANALYZE/NODESIGN [file_spec]
CHECK CALLS [routine]	CHECK CALLS [routine]
COLLAPSE	COLLAPSE
CONVERT LIBRARY [sca_library]	CONVERT LIBRARY [sca_library]
CREATE LIBRARY [sca_library]	CREATE LIBRARY [sca_library]
DELETE LIBRARY [sca_library]	DELETE LIBRARY [sca_library]
DELETE MODULE [sca_module]	DELETE MODULE [sca_module]
DELETE QUERY [query]	DELETE QUERY [query]
EXPAND	EXPAND
EXTRACT MODULE [sca_module]	EXTRACT MODULE [sca_module]
FIND [find_exp]	FIND [find_exp]
GOTO ASSOCIATED DECLARATION [goto_exp]	GOTO DECLARATION/ASSOCIATED [goto_exp]
GOTO CONTEXT DECLARATION [goto_exp]	GOTO DECLARATION /CONTEXT_DEPENDENT [goto_exp]
GOTO DECLARATION [goto_exp]	GOTO DECLARATION/PRIMARY [goto_exp]
GOTO INDICATED DECLARATION	GOTO DECLARATION/INDICATED
GOTO PRIMARY DECLARATION [goto_exp]	GOTO DECLARATION/PRIMARY [goto_exp]
GOTO QUERY [query]	GOTO QUERY [query]
GOTO SOURCE	GOTO SOURCE
INSPECT [inspect_exp]	INSPECT [inspect_exp]
LOAD MODULE [sca_module]	LOAD MODULE [sca_module]
NEXT OCCURRENCE	NEXT OCCURRENCE
NEXT QUERY	NEXT QUERY
NEXT STEP	NEXT STEP
NEXT SYMBOL	NEXT SYMBOL
PREVIOUS OCCURRENCE	PREVIOUS OCCURRENCE
PREVIOUS QUERY	PREVIOUS QUERY
PREVIOUS STEP	PREVIOUS STEP
PREVIOUS SYMBOL	PREVIOUS SYMBOL
RECOVER LIBRARY [sca_library]	VERIFY/RECOVER
REORGANIZE LIBRARY [sca_library]	REORGANIZE LIBRARY [sca_library]
REPORT [report_name]	REPORT [report_name]
SET LIBRARY [sca_library]	SET LIBRARY [sca_library]
SET NOLIBRARY [sca_library]	SET NOLIBRARY [sca_library]
SHOW BRIEF LIBRARY [sca_library]	SHOW BRIEF LIBRARY [sca_library]
SHOW BRIEF MODULE [sca_module]	SHOW MODULE/BRIEF [sca_module]

Portable Command	**VMSLSE Equivalent**
SHOW FULL LIBRARY [sca_library]	SHOW LIBRARY/FULL [sca_library]
SHOW FULL MODULE sca_module	SHOW MODULE/FULL [sca_module]
SHOW LIBRARY [sca_library]	SHOW BRIEF LIBRARY [sca_library]
SHOW MODULE sca_module	SHOW MODULE/BRIEF [sca_module]
SHOW QUERY [query]	SHOW QUERY [query]
VERIFY LIBRARY [sca_library]	VERIFY/NORECOVER [sca_library]
Additional Commands if VMSCMS_ Prefix is Enabled	
CMS [cms_command]	CMS [cms_command]
REPLACE [cms_element]	REPLACE [cms_element]
RESERVE [cms_element]	RESERVE [cms_element]
SET CMS [cms_attributes]	SET CMS [cms_attributes]
SHOW CMS	SHOW CMS
UNRESERVE [cms_element]	UNRESERVE [cms_element]

[1]Indicates that the VMSLSE Equivalent command is incomplete as shown.

Appendix F. Providing 7-Bit Terminal Support for Code Elision

You can use the OpenVMS Terminal Fallback Facility (TFF) to resolve the problem of VT100 terminals displaying unrecognizable characters in place of the double-angle brackets («») displayed on VT200 terminals. The TFF translates the double-angle brackets to single-angle brackets. Have your system manager use the following procedure:

1. Enable TFF by including the following commands in the system startup procedure SYS $MANAGER:SYSTARTUP_V5.COM:

   ```
   $ @SYS$MANAGER:TFF$STARTUP.COM
   ```

2. Add the commands to load the default system fallback and compose sequence tables to the file SYS $MANAGER:TFF$STARTUP.COM.

   For example, to load the necessary fallback and compose-sequence table for use in North America, the system manager would add the following commands:

   ```
   $ RUN SYS$SYSTEM:TFU
   SET LIBRARY SYS$SYSTEM:TFF$MASTER  ! Define the library of tables
   LOAD TABLE ASCII_OVST              ! Load for hardcopy ASCII terminal
   SET DEFAULT_TABLE ASCII            ! Set default to ASCII
   EXIT
   $ EXIT
   ```

After this has been done, you can use the fallback utility. To enable terminal fallback, enter the following command:

```
$ SET TERMINAL/FALLBACK
```

From this point on terminal fallback is enabled. If you want to disable terminal fallback, enter the following command:

```
$ SET TERMINAL/NOFALLBACK
```

If the SOFT_COMPOSE feature is enabled, you must rebind the ERASE PLACEHOLDER and UNERASE PLACEHOLDER keys to something other than Ctrl/K. This is because Ctrl/K is reserved by TFF to signal the initiation of a compose sequence. You might want to use the Ctrl/space and GOLD-Ctrl/space key bindings, but this produces an ASCII NULL, which might cause problems with some communications equipment. It is recommended that SOFT_COMPOSE be disabled unless it is required.

Appendix G. TPU Pattern Style

A new pattern style called Text Processing Utility (TPU) is added to the existing pattern styles VMS and ULTRIX.

For more details on TPU patterns, see the DEC Text Processing Utility Reference Manual.

Pattern styles are used in the following:

- SEARCH/PATTERN command (VMS command language)
- SUBSTITUTE/PATTERN command (VMS command language)
- PATTERN SEARCH command (Portable command language)
- PATTERN SUBSTITUTE command (Portable command language)
- PATTERN EXACT SUBSTITUTE command (Portable command language)
- Search dialog box (Search / Search ...)
- Substitute dialog box (Search / Substitute ...)

The usage of the TPU pattern style is similar to the existing pattern styles. The main advantages of the TPU style are as follows:

- Direct access to the powerful TPU pattern facility (the existing pattern styles use the facility indirectly).
- The ability to make a substitution that is a function of the pattern found.
- The ability to develop VSI Test Manager for OpenVMS (DTM) user defined filters interactively.

Most of the examples in the following sections are given using the VMS command language and assume that the search options are set to the TPU pattern style and case insensitive searching. The search direction is assumed to be forward and the cursor is assumed to be positioned before the example text. The search and replace parameters in the VMS command language examples can be used unchanged in the equivalent Portable command language commands.

G.1. User Interface

Select the TPU pattern style using one of the following options:

- SET SEARCH PATTERN=TPU (vms command language)
- SET SEARCH PATTERN TPU (portable command language)
- Select TPU pattern on the Search Attributes window

The only other user interface change is the form of the search and replace string parameters when the TPU pattern style is selected fora pattern search or substitution.

The search string parameter is a TPU expression that must evaluate to a TPU pattern and the replace string parameter is a TPU expression that must evaluate to a TPU string.

Following are the examples in both the command languages. The first two examples search for 'abc' or 'def' and the last two examples substitute all occurrences of 'abc' or 'def' by 'ghi':

```
SEARCH/PATTERN  "'abc' | 'def'"
PATTERN SEARCH  "'abc' | 'def'"

SUBSTITUTE/PATTERN/ALL   "'abc' | 'def'"  "'ghi'"
PATTERN SUBSTITUTE  "'abc' | 'def'"  "'ghi'" ALL
```

In the above examples 'abc', 'def' and 'ghi' are TPU strings and | is the TPU pattern alternation operator.

The outermost quotes in the above example must be omitted if the parameters are prompted for or if a dialog box is used.

G.2. Partial Pattern Assignment Variables

Partial pattern assignment variables allow a substitution to be a function of the found pattern.

For example, the following command replaces the date format 'yyyy/mm/dd' to 'dd/mm/yyyy':

```
SUBSTITUTE/PATTERN -
  "(_year@_v1)+'/'+(_month@_v2)+'/'+(_day@_v3)" -
  "str(_v3)+'/'+str(_v2)+'/'+ str(_v1)"

when applied to:  1998/04/21  generates:  21/04/1998
```

In the above example, _year, _month and _day are TPU variables holding patterns that match the year, month and day parts of a date. For more information on setting up the variables, refer to Section G.6, Pattern Variables.

In addition, @ is the TPU partial pattern assignment operator, and _v1,_v2 and _v3 are partial pattern assignment variables that are set to the found year, date and day.

A partial pattern assignment variable holds a TPU range. When used in the replacement string must be converted to a string using the TPU procedure STR.

For example, the following command prefixes XYZ_ to any line that starts with any three characters from ABCDEFGHI:

```
SUBSTITUTE/PATTERN/ALL -
   "LINE_BEGIN + (ANY('ABCDEFGHI',3)@_v1)" -
   "'XYZ_'+ str(_v1)"

when applied to: abc        generates: XYZ_abc
                 012                   012
                 defghi                XYZ_defghi
```

In the above example, LINE_BEGIN is a TPU keyword that matches the beginning of a line and ANY is a TPU pattern procedure that matches a specified number of characters from a specified set of characters.

G.3. New Line

A new line is generated for each line feed character in the replacement string. A line feed character can be introduced by means of the TPU procedure ASCII with the value 10 as a parameter.

For example, to replace numbers at the end of lines with the string 'xxx' (a line feed is necessary because the search pattern includes the end of the line):

```
SUBSTITUTE/PATTERN/ALL -
   "_n + LINE_END" -
   "'xxx' + ASCII(10)"
when applied to: 123 456    generates: 123 xxx
                 789                   xxx
```

In the above example, LINE_END is a TPU keyword that matches the end of a line and _n is TPU variable holding a pattern that matches a number.

When TPU procedure STR converts a partial pattern assignment variable to a string, an optional second parameter is set to ASCII(10) to cause any end of lines in the range described by the variable to be converted to line feed characters (without the parameter they are represented by the null string). For example:

```
SUBSTITUTE/PATTERN/ALL -
   "(LINE_BEGIN + _n + LINE_END + _n + LINE_END)@_v1" -
   "STR(_v1, ASCII(10)) + STR(_v1, ASCII(10))"

when applied to: 123        generates: 123
                 456                   456
                                       123
                                       456
```

Carriage return characters adjacent to line feed characters in the replacement string are ignored.

G.4. Errors

The search and replace strings are TPU expressions that must be evaluated. During the process of evaluation, there might be various TPU compilation/evaluation errors messages generated.

Two new error messages are added for invalid search or replace parameters:

```
Error in search pattern
Error in replacement string
```

These messages are normally preceded by various TPU error messages. For example, the search string "'aaa' + bbb" results in the following error messages:

```
Undefined procedure call BBB
Operand combination STRING + INTEGER unsupported
Error in search pattern
```

G.5. Global Variables

Partial pattern assignment variables and pattern variables (such as '_year' in the preceding examples) must be global and not clash with any TPU global variables used by LSE. This can be achieved by starting any such variable names with an underscore character.

G.6. Pattern Variables

Any complicated search or substitution is likely to need various pattern variables to be already set up. This can be achieved in various ways.

The definitions can be set up by issuing DO/TPU commands, for example:

```
DO/TPU "_digits:='0123456789'"
DO/TPU "_digit:=any(_digits)"
DO/TPU "_year:=any(_digits,4)"
DO/TPU "_month:=any('01',1)+_digit"
DO/TPU "_day:=any('0123',1)+_digit"
DO/TPU "_n:=span(_digits)"
```

The file LSE$PATTERNS.TPU in the LSE$EXAMPLE directory contains examples of patterns, which can be added to LSE by means of the following commands:

```
GOTO FILE LSE$EXAMPLE:LSE$PATTERNS.TPU
EXTEND *
DO/TPU "LSE$PATTERNS_MODULE_INIT"
```

G.7. Use for Developing DTM User Filters

The user defined filters global replace feature introduced in DTM Version 4.0 can be simulated using the SUBSTITUTE/PATTERN/ALL command. This allows DTM user defined filters to be developed interactively using LSE.

For example, to replace any numbers at the end of lines with the string 'xxx':

```
global_replace(
                    _n + LINE_END,
                    'xxx' + ASCII(10),
                    NO_EXACT,
                    OFF,
                    ON);
```

The LSE equivalent (assuming that the current search attributes are equivalent to NO_EXACT) is:

```
SUBSTITUTE/PATTERN/ALL -
   "_n + LINE_END" -
   "'xxx' + ASCII(10)"
```

The LSE equivalent of the pattern to replace parameter (first parameter of the global_replace routine) is the same except that the parameter has to be in quotes.

The LSE equivalent of the replacement string parameter (second parameter) is the same if the evaluate replacement parameter (fourth parameter) is set to ON. If the evaluate replacement parameter is set to OFF the parameter must be in quotes.

The LSE equivalent of the search mode parameter (third parameter) is the setting of the search options (set by the SET SEARCH command).

LSE does not have equivalents of the evaluate replacement parameter (fourth parameter) or the convert linefeeds parameter (fifth parameter). It always evaluates the replacement string parameter and it always converts linefeed characters (and ignores adjacent carriage return characters).