

C/C++ Productivity Tools for OS/390



Editor

Release 1.0

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 281.

First Edition (September 1999)

This edition applies to C/C++ Productivity Tools for OS/390 Release 1.0, program number 5655-B85 and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable system bibliography for current information on these products.

Order publications through your IBM representative or through the IBM branch office serving your locality.

© **Copyright International Business Machines Corporation 1999. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	v
Who should read this book	v
Conventions used in this book	v
Related information	v
How to send your comments	v

Chapter 1. Introducing the Editor	1
Introducing the Editor	1
Editor commands and parameters	1
Editor Customization	2
Using alternate editor profiles	2
Using parsers	3

Chapter 2. Working with text in the Editor	5
Starting the editor	5
Entering and editing text	5
Undoing changes	7
Marking and manipulating blocks of text	7
Marking blocks of text	7
Unmarking a block of text	8
Manipulating blocks of text	9
Changing the default marking mode	9

Chapter 3. Finding text or locations in a file	13
Finding a specific line in a file	13
Using location marks in a file.	13
Finding text	14
Finding and replacing text.	15

Chapter 4. Managing files in the Editor	17
Creating a new file	17
Opening an existing File	17
Saving a file	18
Embedding another file into the current document	18
Working with multiple documents	19

Chapter 5. Working with Editor commands and settings	21
Issuing editor commands	21
Changing editor tab settings	21
Changing the base editor font	22
Customizing the Keyboard	22
Changing text and background color palettes	23

Chapter 6. Reference	25
Default editor commands	25
action command	26
add command	26
block command	27
compare command	30
delete command	32

deleteText command.	33
expandAll command	33
findText command	34
get command	37
help command.	38
input command	39
insert command	40
insertShow command	40
insertText command	41
load command.	42
locate command	42
parse command	44
print command	44
processPrefix command.	46
query command	47
replaceText command	49
resequence command	50
save command.	50
screenShow command	52
set command	52
undo command	54
updateProfile command	55
Editor parameters	58
actionClass parameter	63
actionKey parameter.	64
actionKeyText parameter	65
actions parameter.	65
baseProfile parameter	66
beep parameter	66
block.defaultType parameter	67
changes parameter	69
class parameter	70
classes parameter	71
commandClass parameter	71
commandLine parameter	72
commands parameter	74
compare.ignoreCase parameter	74
compare.ignoreLeadingBlanks parameter	76
compare.ignoreTrailingBlanks parameter	78
compare.ignoreAllBlanks parameter	80
cursorRow parameter	82
default parameter.	83
dirty parameter	85
displayPosition parameter	85
documentId parameter	86
element parameter	86
elementClasses parameter	87
elements parameter	88
emphasisLength parameter	88
excludedClasses parameter	89
expandHide parameter	90
expandHideAreaWidth parameter	92
expandTabs parameter	92
expanded parameter.	94
fields parameter	95
findText.asis parameter	96

findText.block parameter	98	recording parameter	175
findText.columns parameter	100	rowHeight parameter	175
findText.emphasis parameter	102	rows parameter	176
findText.endColumn parameter	104	save.textLimit parameter	176
findText.findText parameter	106	save.trim parameter	178
findText.mark parameter	108	scroll parameter	179
findText.regularExpression parameter	110	sequenceNumber parameter	180
findText.replaceText parameter	112	sequenceNumberStyle parameter	181
findText.startColumn parameter	114	sequenceNumbers parameter	182
findText.wrap parameter	116	show parameter	184
font parameter	118	status parameter	184
forceAllVisible parameter	120	statusLine parameter	185
forceVisible parameter	120	style parameter	186
headerMark parameter	121	styleAttributes parameter	187
help.configuration parameter	122	tabs parameter	189
help.homepage parameter	123	text parameter	191
help.location parameter	123	textAreaWidth parameter	191
hex parameter	124	textWidth parameter	192
hideSequenceNumbers parameter	125	topExpanded parameter	193
inPrefix parameter	126	updateProfile.baseProfile parameter	194
includedClasses parameter	127	updateProfile.extensions parameter	196
insertMode parameter	128	updateProfile.noParser parameter	197
install parameter	129	updateProfile.palette parameter	199
installProfile parameter	131	updateProfile.paletteAttributes parameter	201
keyAction parameter	133	updateProfile.palettes parameter	203
keys parameter	136	updateProfile.parser parameter	204
length parameter	137	updateProfile.parserAssociation parameter	206
line parameter	137	updateProfile.parserClass parameter	208
lineNumbers parameter	138	updateProfile.parsers parameter	210
lines parameter	139	updateProfile.userActions parameter	211
maintainSequenceNumbers parameter	140	updateProfile.userCommands parameter	213
mark parameter	141	updateProfile.userKeyActions parameter	215
markExcluded parameter	143	updateProfile.userMouseActions parameter	217
markExcludedHeader parameter	144	updateProfile.userProfile parameter	219
markHighlight parameter	145	visible parameter	221
markId parameter	146	Default editor actions	222
markIncluded parameter	147	brief base profile	231
markProtect parameter	148	Key Settings	231
markStyle parameter	149	Mouse Event Settings	236
messageLine parameter	150	epm base profile	238
messageText parameter	151	Key Settings	238
mouseAction parameter	151	Mouse Event Settings	243
mouseEvents parameter	153	ispf base profile	244
name parameter	154	Key Settings	244
palette parameter	154	Mouse Event Settings	249
parser parameter	155	Prefix Commands.	250
pixelPosition parameter	155	lpex base profile	254
popup parameter	156	Key Settings	254
position parameter	158	Mouse Event Settings	258
prefixAreaWidth parameter	158	seu base profile	260
prefixPosition parameter	159	Key Settings	260
prefixProtect parameter	160	Mouse Event Settings	264
prefixText parameter	161	Prefix Commands.	265
print.bottomMargin parameter	162	xedit base profile	271
print.font parameter	163	Key Settings	271
print.leftMargin parameter	165	Mouse Event Settings	276
print.lineNumbers parameter	167	Prefix Commands.	277
print.rightMargin parameter	169		
print.tokenized parameter	171		
print.topMargin parameter	172		
readonly parameter	174		
		Notices	281
		Trademarks and service marks	283

About this book

Editor introduces you to the Editor and provides information about how to edit an OS/390 C and C++ application.

Who should read this book

Editor is intended for application programmers who want to edit C and C++ applications on OS/390 and want a workstation editor to enhance their existing familiar host environment. For these users, this document introduces the editor and shows how to use it.

Conventions used in this book

The following conventions distinguish different text styles within this book:

plain	Window titles, folder names, icon names, and method names.
monospace	Programming examples, user input at the command line prompt or into an entry field, directory paths.
bold	Menu choices and menu names, labels for push buttons, check boxes, radio buttons, group-box controls, drop-down list boxes, combination-boxes, notebook tabs, and entry fields.
<i>italics</i>	Programming keywords and variables, and titles of documents.

Related information

For information on OS/390 C/C++ related features, news and Web sites, add this Web site to your browser's bookmark list:

<http://www.ibm.com/software/ad/c390>

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this book or any other C/C++ Productivity Tools documentation, send your comments by e-mail to torrcf@ca.ibm.com. Be sure to include the name of the book, the document number of the book, the version of C/C++ Productivity Tools, and, if applicable, the specific location of the information on which you are commenting (for example, a page number or a table number).

Chapter 1. Introducing the Editor

Introducing the Editor

The editor can be used to create and edit many kinds of files, including program source files, documentation, and data files.

In addition to basic editing functions, the editor also offers the following features:

- Language parsing uses automatic indenting, color, and text effects to emphasize different parts of your source program, such as programming language keywords, logic structures, comment lines, and arithmetic operators.
- Location marking facilities let you define *bookmarks* that let you move quickly from one location to another within your source file.
- Elaborate search facilities let you specify the precise scope of the search. In addition to locating specific character strings, other search facilities let you find specific locations within your file.
- Multiple editing views let you have more than one file open for editing at a time.
- Multiple editing windows let you open multiple views of the same file for editing, letting you view different parts of the file at the same time. Changes to a file in any one view are also reflected in other views containing that same file.
- Keystroke recording facilities let you *record* sets of keystrokes which you can later replay.

RELATED CONCEPTS

“Editor commands and parameters”

“Working with multiple documents” on page 19

“Editor Customization” on page 2

“Editor commands and parameters”

RELATED TASKS

“Entering and editing text” on page 5

“Working with multiple documents” on page 19

Editor commands and parameters

The editor is fully programmable through the use of an extensive command and parameter set. You can use commands and parameters to customize your editor window, search for or change text in your document, or perform many other functions. You can find more information about using commands and parameters in the related readings below.

RELATED TASKS

“Issuing editor commands” on page 21

RELATED REFERENCES

“Default editor commands” on page 25

“Editor parameters” on page 58

Editor Customization

You can customize editor appearance and function to suit individual preferences or project needs. Simple customizations can be made with pulldown menu selections or by setting Editor parameters. You can also create macros and load profiles to handle more elaborate customizations.

See the topics below for instructions on how to perform common Editor customizations.

RELATED TASKS

“Changing editor tab settings” on page 21

“Changing the base editor font” on page 22

“Customizing the Keyboard” on page 22

“Changing text and background color palettes” on page 23

Using alternate editor profiles

You can configure the editor to adopt the keyboard and command personalities of many popular editors.

Most editor profiles differ only in the keys and commands used to perform various editor tasks. Some editor profiles, listed below, also add a prefix information and command area at the start of each line.

- ISPF
- SEU
- XEDIT

The editor recognizes prefix commands used by these editor profiles. Depending on which profile you are using, you can enter SEU, XEDIT, or ISPF commands when the prefix area is active. You can also create your own commands by binding any editor macro or command to your own line commands.

The prefix area appears on the left side of the Editor pane, and may display line numbers and/or the date. Line numbers are maintained by the editor.

Refer to related readings below for more information on using prefix area commands for specific editor personalities.

RELATED REFERENCES

“brief base profile” on page 231
“epm base profile” on page 238
“ispf base profile” on page 244
“lpex base profile” on page 254
“seu base profile” on page 260
“xedit base profile” on page 271

Using parsers

A parser is an editor command that acts interactively on a document to improve the presentation of the data in that document. A parser uses colors and fonts to highlight different items in a programming language document. For example, language keywords are highlighted in one color, variable names in another, and string literals in yet another. The original indentation of the program code is maintained. The editor comes with a selection of parsers for common programming languages.

Invoking a Parser

When you open a file, the editor looks to see if the file name extension of that file is associated with a parser. For example, the parser comes configured to recognize and parse C program files. If you open a file called **sample.c**, the editor invokes the C parser. If you open a file called **sample.asm**, the editor invokes the hlsam parser.

Typically, the parser will:

- Set default fonts and colors
- Color and emphasize the data being displayed
- Set up special actions for keys
- Set up language-specific additions to the editing pane pop-up menu.

About Elements and Classes

In a programming language document, each line is an element. A class definition describes the type of data the element contains. Each element may contain more than one class. The elements displayed below include **CODE** and **COMMENT** classes

Note: Class names are chosen by the writer of the parser.

	CODE Class	COMMENT Class
Lines of C code, an element	if (x == "test")	/*test for x*/

Using a Parser's Editing Pane Pop-up Menu

The selections in the editing pane's pop-up menu specify which classes are displayed in the document. For example, you can select the **Functions** selection from the pop-up menu to display only function headers in the C document.

Live Parsing

The editor monitors and records all the changes that you make to a document. As you complete each line, the editor examines that line for defined class elements. For example, in a C program, the parser recognizes the text between an open

comment marker (/*) and a close comment marker (*/) as being comments, and displays those comments in the color and font specified for the comment class. Other text, even if on the same line as a comment, is displayed according to class.

Chapter 2. Working with text in the Editor

Starting the editor

To start the editor, do any of the following:

- At a command line prompt, type

```
iedit filename.ext
```

where *filename.ext* is optional and represents the name of the file you want to edit.

- If you have an icon for the editor installed on your desktop, double-click on the icon.

RELATED TASKS

“Opening an existing File” on page 17

Entering and editing text

Entering Text

- Click in the working area of the Editor pane and begin typing.
- To type on the command line, click in the command line area or press the Esc key and then begin typing.

Replacing and Inserting Text

- The status line indicates which mode the editor is in. When the editor is in *Replace* mode, the cursor appears as a solid block, one character width in size. Text overlaid by this block will be replaced by any new text that you type. When the editor is in *Insert* mode, the cursor appears as a thin vertical line. Any new text that you type is inserted into the file at the cursor position, and existing text to the right of the cursor will be shifted to the right.
- To toggle back and forth between the two modes, press the Insert key.

Deleting Text

- There are many ways to delete unwanted text. To delete single characters, do one of the following:
 - Press Delete. If you are in *Insert* mode, the character to the right of the cursor is deleted; if you are in *Replace* mode, the character that is overlaid by the cursor is deleted. Text to the left of the cursor moves right to fill the resulting gap.
 - Press Backspace. The character to the left of the cursor is deleted. Text to the right of or overlaid by the cursor moves left to fill the resulting gap.
- To delete all the text between the cursor and the end of the current line, press Ctrl+Delete.
- To delete a complete line, move the cursor to the unwanted line and press Ctrl+Backspace.

Moving the Cursor

- Position the mouse pointer where you want the cursor to be, and click mouse button 1, or,
- Press Up, Down, Left, or Right arrow, or,
- Press Home to move the cursor to the beginning of a line, or End to move it to the end of a line, or,
- Press Ctrl+Left arrow to move the cursor one word left, or Ctrl+Right arrow to move it one word right, or,
- Press Page Up or Page Down to move the cursor up or down one window at a time, or,
- Press Ctrl+Home to move the cursor to the beginning of the file, or Ctrl+End to move it to the end of the file, or,
- Press Ctrl+J to return the cursor to the place in the file where you last entered text.

Tip! If you inadvertently select text by dragging the mouse, deselect it by pressing Alt+U, or, click the right mouse button and choose **Deselect** from the pop-up menu.

Inserting a Blank Line

- Move the cursor to any point on a line and press Ctrl+Enter, or,
- Move the cursor to the end of a line and press Enter.

A blank line is created after the current line, and the cursor moves to the first column position of this new line.

Splitting and Joining Lines of Text

- To split a line:
 - Place the cursor where you want the break to occur, and press either Alt+S or Enter. Pressing Alt+S leaves the cursor at its current position; pressing Enter moves the cursor to the beginning of the new line.

When you split a line, all the text to the right of the cursor is moved down to a new line.

- To join two lines, do one of the following:
 - If the cursor is at the end of a line, press Delete to join the current line with the next line.
 - If the cursor is at the beginning of a line, press Backspace to join the current line with the previous line.
 - With the cursor anywhere on a line, press Alt+J to join the current line with the next line together.

RELATED TASKS

“Undoing changes” on page 7

Undoing changes

The editor records each set of changes you make to a file in the editor window. The number of changes made since the last file save is displayed on the status line.

If you want to undo a set of changes to a file, do the following:

1. Select **Edit** from the editor window menu bar.
2. Select **Undo**.
3. Repeat the above steps until all unwanted changes are undone.

You can also cancel the effects of an undo operation by doing the following.

1. Select **Edit** from the editor window menu bar.
2. Select **Redo**.
3. Repeat the above steps until all the effects of all unwanted undo operations are cancelled.

RELATED TASKS

“Entering and editing text” on page 5

Marking and manipulating blocks of text

The editor offers facilities that let you mark blocks of text in various ways to accomplish different results. Once you have marked a block of text, you can manipulate the marked block with either standard operating-system clipboard facilities or the editor’s own block manipulation facilities.

RELATED TASKS

“Marking blocks of text”

“Unmarking a block of text” on page 8

“Manipulating blocks of text” on page 9

“Changing the default marking mode” on page 9

Marking blocks of text

There are many ways of marking text in a file that is being edited. Some are described below.

Marking a Block of Text with the Mouse

1. Position the mouse pointer over the character that is to start the block.
2. Right click and hold the mouse button.
3. Drag the mouse pointer to the to the end of the block of text you want and release the mouse button. The block is now marked according to the default blocking mode in effect.

Marking a Block of Text using the Keyboard

1. Use the cursor keys to move the cursor to the character that will start the block.

2. Press Alt+B.
3. Move the cursor to the end of the block of text.
4. Press Alt+B again. Text between the two points is marked.

Marking an Entire Line of Text

1. Place the cursor anywhere on the line to be marked.
2. Select **Edit** from the main menu bar.
3. Select **Select**.
4. Select **Line**. The line is marked.

To mark all lines between the line selected above and another line inclusive:

1. Move the cursor to the last line of the group of lines you want to select
2. Use the steps described above to mark that line. All lines between the first and second lines selected are marked.

Marking a Rectangular Area of Text

1. Place the cursor on the character position defining one corner of the area you want to mark.
2. Select **Edit** from the main menu bar.
3. Select **Select**.
4. Select **Rectangle**. The first corner of the rectangle is marked.
5. Place the cursor on the character position defining the opposite corner of the rectangular area of text you want to mark, and repeat steps 2 to 4 above. The rectangular area between the two corners is marked.

RELATED TASKS

- “Marking and manipulating blocks of text” on page 7
- “Unmarking a block of text”
- “Manipulating blocks of text” on page 9
- “Changing the default marking mode” on page 9

Unmarking a block of text

To unmark a block of text, use any of the following methods:

- Select **Edit** from the main menu bar, then select **Deselect**, or,
- Right-click the mouse and select **Deselect**, or,
- Press Alt+U.

RELATED TASKS

- “Marking and manipulating blocks of text” on page 7
- “Marking blocks of text” on page 7
- “Manipulating blocks of text” on page 9
- “Changing the default marking mode” on page 9

Manipulating blocks of text

After you have marked a block of text, you can manipulate it in any of the ways described below.

Copying a Marked Block

1. Move the cursor to the location in the file where the blocked text will be copied.
2. Select **Selected** from the main menu bar
3. Select **Copy selected text**. A copy of the marked block is inserted at the cursor position.
Alternatively, position the cursor and then press Alt+C to copy a selected block.

Deleting a Marked Block

1. Select **Selected** from the main menu bar
2. Select **Delete selected text** to delete the marked block.
Alternatively, press Alt+D to delete a marked block.

Moving a Marked Block

1. Move the cursor to the desired location.
2. Select **Selected** from the main menu bar
3. Select one of the following:
 - a. **Moveselected text** - The block is placed below the cursor. Existing text below the cursor shifts downward to allow room for the moved block of text.
 - b. **Overlayselected text** - The block is placed below the cursor, overwriting any existing text.

Alternatively, position the cursor and then press Alt+M, or use Alt+Z if you want to place the block over existing text.

RELATED TASKS

“Marking and manipulating blocks of text” on page 7
“Marking blocks of text” on page 7
“Unmarking a block of text” on page 8
“Changing the default marking mode”

Changing the default marking mode

You can change the default marking mode to one of the following choices:

stream	Default. This is the marking mode commonly used in most text editors and word processors. Text is selected on a character to character basis.
character	Unlike the other marking modes, the cursor cannot be moved to another part of the file without deselecting a block of selected text. Once text is selected, the next keyboard action will usually affect the entire selection of text. For example, pressing a keyboard character will replace the entire selected text with that character, and pressing the Delete key will remove the entire selected block. Similar to stream, except that the cursor is <i>not</i> coupled with the marked text selection. You can mark a block of text with the mouse, and then click the mouse at another location in the file without unmarking the block of text already selected.
element	Whole elements are marked. The cursor is not coupled to the marked text selection.
rectangle	Rectangular areas can be marked starting at any character position on a line and extending over as many lines as needed. The cursor is not coupled to the marked text selection.

To change the default block marking mode for new files or files already open for editing:

1. Select **File** from the main menu bar.
2. Select **Preferences**. The Application Preferences dialog appears.
3. Expand the **Editor** preferences tree by clicking on the + sign.
4. Select **Block** from the expanded list of **Editor** preference settings. The Block pane appears in the Application Preferences dialog.
5. Select one of the available block types from the **Default block type** selection list.
6. Click **Apply** to apply the selected block type to the editor.
7. Click **OK** to close the Application Preferences dialog.

To change the default block marking mode only for the current file:

1. Press **Esc** to open the editor command line.
2. Enter the following command on the editor command line:

```
set block.defaultType mode
```

where *mode* is one of the marking modes described above.

To determine the current default block marking mode for the current file:

1. Press **Esc** to open the editor command line.
2. Enter the following command on the editor command line:

```
query current.block.defaultType
```
3. The current blocking mode is returned in the message area below the editing area.

RELATED TASKS

- “Marking and manipulating blocks of text” on page 7
- “Marking blocks of text” on page 7
- “Unmarking a block of text” on page 8
- “Manipulating blocks of text” on page 9

Chapter 3. Finding text or locations in a file

Finding a specific line in a file

A quick way to find a specific line in a file is to use the **Locate Line** facility.

1. Press **Ctrl+L**. The **Line number** input area appears.
2. Enter the number of the line you want to go to and press enter. The **Line number** input area closes and the cursor moves to the chosen line.

You can also use the editor **locate** command to find a specific line. For example:

1. Press **Esc** to open the editor command line area.
2. Enter the following editor command in the command line area:

```
locate line n
```

where *n* is the number of the line you want to go to.

3. Press **Esc** again to leave the command line input area. The cursor is now at the chosen line.

Note: If you try to go to a line number that is greater than the number of lines in the file, the cursor moves to the beginning of the last line in the file.

RELATED TASKS

Finding a location mark in a file
“Finding text” on page 14
“Finding and replacing text” on page 15

RELATED REFERENCES

“locate command” on page 42 command

Using location marks in a file

Location marks function like bookmarks. You place them in your document and use them to easily return to *marked* locations in your file.

Creating and Naming a Location Mark

To create and name a location mark in your document, do the following:

1. Place the cursor at the position to be marked.
2. Select **Edit** from the main menu bar.
3. Select **Mark**.
4. Select **Name a mark**.
5. Enter the name of the new mark in the text entry field of the **Name mark** input area.
6. Press **Enter** to save the location mark.

Finding a Named Location Mark

To move the cursor to a named mark, do the following:

1. Select **Edit** from the main menu bar.
2. Select **Find other**.
3. Select **Find mark**.
4. The dialog box **Find mark** input area appears. Select the name of the mark you want to move to. You can use the **Up** and **Down** arrow keys to scroll through existing marks.
5. Press **Enter** to move to the location of the chosen mark.

Using a Quick Location Mark

To mark one cursor position quickly, you can use the **Quick Mark** feature as follows:

1. Move the cursor to the position where you want to set a mark.
2. Select **Edit** from the main menu bar.
3. Select **Mark**.
4. Select **Set quick mark**. A quick-mark is created at the chosen position in your file.

To find a quick-mark, do the following:

1. Select **Edit** from the main menu bar.
2. Select **Find other**.
3. Select **Find quick mark**. The cursor moves to the position marked by the quick mark

You can have only one quick mark per document view. Setting a new quick mark will cause a previous quick mark to be lost.

You can also find a quick mark using the procedure described in **Finding a Named Location Mark** above. The quick mark will appear in the list of named marks as **@QUICK**.

RELATED TASKS

“Finding text”



“Finding and replacing text” on page 15

Finding text

To search for an item in your document or file view, first do the following to open the **Find** input area:

1. Select **Edit** from the main menu.
2. Select **Find**. The **Find** input area appears at the bottom of the Editor pane.

Specify the item to be searched for by doing the following:

1. Type a search item in the **Find** text entry area. This can be a word, a partial word, or a sequence of such.
2. Select your desired options.
3. Click **Next** or **Previous**. If the entered text or pattern is found, the cursor moves to either the next or previous occurrence of the search item, according to your chosen search direction.
4. Alternatively, click **All**. Only lines containing the text in the **Find** text entry field are shown, and all other lines are hidden from view. If you have enabled the **Expand/hide area** controls, a  symbol appears to represent parts of your file where lines have been hidden from view. Clicking on the  symbol displays those hidden lines. To show all lines again, select **Edit->Show all** from the editor menu bar.
5. Close the **Find** text entry area by pressing **Esc**.

RELATED TASKS



Finding a location mark in a file
 “Finding and replacing text”

Finding and replacing text

To search for an item in your document or file view, first do the following to open the **Find/Replace** input area:

1. Select **Edit** from the main menu.
2. Select **Find/Replace**. The **Find/Replace** input area appears at the bottom of the Editor pane.

Specify the item to be searched for and replaced by doing the following:

1. Type a search item in the **Find** text entry area. This can be a word, a partial word, or a sequence of such.
2. Type the replacement word in the **Replace** text entry field.
3. Select your desired options.
4. Click **Next** or **Previous**. If the entered text or pattern is found, the cursor moves to either the next or previous occurrence of the search item, according to your chosen search direction, and locates the found text according to your selections. If you want to change the found text, click **Replace**. Repeat this step to find and replace multiple occurrences of the chosen text in your file.
5. Alternatively, click **Replace all**. All occurrences of the chosen text are replaced and only lines where replacement has occurred are shown in the Editor pane. All other lines are hidden from view. If you have enabled the **Expand/hide area** controls, a  symbol appears to represent parts of your file where lines have been hidden from view. Clicking on the  symbol displays those hidden lines. To show all lines again, select **Edit->Show all** from the editor menu bar.
6. Close the **Find/Replace** text entry area by pressing **Esc**.
- 7.

RELATED TASKS

“Finding text” on page 14

Finding a location mark in a file

Chapter 4. Managing files in the Editor

Creating a new file

You can open a new file by doing the following:

1. Select **File** from the main menu bar.
2. Select **New file**.
3. An empty, unnamed file appears in the Editor pane.

RELATED TASKS

“Opening an existing File”

“Saving a file” on page 18

“Embedding another file into the current document” on page 18

“Working with multiple documents” on page 19

Opening an existing File

You can open an existing file by doing any of the following:

- Drag a file icon from a file manager window onto an editor icon.
- At a command line prompt, type

```
iedit filename.ext
```

where *filename.ext* is the name of the file you want to edit.

- If the editor is already running and the file you want to open has been recently edited, do the following:
 1. Select the **Recently Opened Files** list in the File Selector pane.
 2. Double-click on the name of a file to open that file for editing.
- If the editor is already running and the file you want to open has not been recently edited, do either of the following:
 1. Select the directory tree list in the File Selector pane.
 2. Traverse through the directory tree to find the file you want to open.
 3. Double-click on the name of a file to open that file for editing.

or,

1. Select **File** from the editor menu bar.
2. Select **Open file**.
3. Use the file and directory selection lists to select the file you want to open.
4. Click **OK** to open the selected file for editing.

Note: The editor will not try to open a file that is already loaded in the editor.

RELATED TASKS

“Creating a new file” on page 17

“Saving a file”

“Embedding another file into the current document”

“Working with multiple documents” on page 19

Saving a file

To save your work, do any of the following:

1. To save one file:
 - a. If you have more than one file open for editing, select the file that contains the work you want to save.
 - b. Select **File** from the editor menu bar.
 - c. Select **Save file**.
 - d. If you are saving an existing file, the file is saved under its current name. If you are saving a new file, the Save As dialog box appears. Enter a new name for the file and click OK. The new file is saved under this name.
2. To save one file to a different file name:
 - a. If you have more than one file open for editing, select the file that contains the work you want to save.
 - b. Select **File** from the editor menu bar.
 - c. Select **Save as**. The **Save as** dialog box appears.
 - d. Enter a new name for the file and click OK. The new file is saved under this name.
3. To save changes to all files you are currently working with, do the following while in any open editor window:
 - a. Select **File** from the editor menu bar.
 - b. Select **Save all open files**. Existing files are saved under their current file names. For each new, unnamed file saved, the Save as dialog box appears. Enter the new name of the file and click OK to save the new file under that name.

RELATED TASKS

“Creating a new file” on page 17

“Opening an existing File” on page 17

“Embedding another file into the current document”

“Working with multiple documents” on page 19

Embedding another file into the current document

The **Get** operation embeds another file into the file currently open in the editor. This operation does not load any new profiles along with the file being inserted. Profiles already loaded remain in effect and are applied to the file being inserted.

Embed another file into your document by doing the following:

1. Move the cursor anywhere on the line above where the inserted text is to appear.
2. Select **File** from the editor menu bar.
3. Select **Get file**. The **Get** window appears.
4. Select the desired file and click **OK**. The inserted text appears on the line(s) after the cursor position.

RELATED TASKS

“Creating a new file” on page 17
“Opening an existing File” on page 17
“Saving a file” on page 18
“Working with multiple documents”

Working with multiple documents

The editor lets you have several documents (file views) open for editing at the same time. These views can be of different documents, or different views of the same document. Displaying multiple views of the same document can help you perform functions like cutting and pasting from one section to another. Any changes you make to one file view are automatically updated in all other views of that document. You can also carry out functions like cutting and pasting between views of different documents. You can open multiple views of a file already open for editing by doing the following:

1. Select **Window** from the editor menu bar.
2. Select **New Editor**. A new, empty editor frame appears.
3. Use the **Open Files** selection list in the File Selector pane of the new editor frame to select and open a file already open for editing in another editor frame.

RELATED TASKS

“Opening an existing File” on page 17

Chapter 5. Working with Editor commands and settings

Issuing editor commands

You can use editor commands to customize your editing environment, search for or change text in your document, or perform many other functions.

To issue an editor command:

1. Press **Esc** to bring up the editor command line below the editing area.
2. Type your editor command on the command line, then press **Enter** to perform the command. For example, entering the command:

```
add 1
```

will add a new line into the file

To recall a previously-used command:

1. While in the command line, press **Up** or **Down**. Previously used commands will appear in the message window.
2. Select the command you want to reuse, then press **Enter**.

RELATED REFERENCES

"Default editor commands" on page 25

Changing editor tab settings

You can easily change the editor tab stops to suit the requirements of the document you are working on.

To change the tab stops, you must first open the Tabs pane in the Application Preference dialog:

1. Select **File** from the editor menu bar.
2. Select **Preferences**. The Application Preferences dialog appear.
3. In the Application Preferences dialog, expand the **Editor** preferences tree by clicking on the + beside **Editor**.
4. Select **Tabs** from the expanded list of **Editor** preferences. The Tabs preference pane appears.

Within this pane, do the following:

1. In the **Tab stops** field, specify specific column positions in which you want tab stops to appear. For example, if you want tab stops at columns 1, 3, and 5, enter the following:

```
1 3 5
```

2. Specify a tab increment span in the **Tab increment** field. For example, a value of 4 would cause a tab stop to be placed at every fourth character position following the last tab stop defined in the **Tab stops** field.

3. If you want the editor to expand tab characters into the appropriate number of hard space characters, select the **Expand tab characters** check box.
4. Click **Apply** to apply the tab settings to the editor, then click **OK** pushbutton to close the Application Preferences dialog.

RELATED CONCEPTS

“Editor Customization” on page 2

Changing the base editor font

To customize the base font used by the editor, you must first open the Font pane in the Application Preference dialog:

1. Select **File** from the editor menu bar.
2. Select **Preferences**. The Application Preferences dialog appear.
3. In the Application Preferences dialog, expand the **Editor** preferences tree by clicking on the + beside **Editor**.
4. Select **Font** from the expanded list of **Editor** preferences. The Font preference pane appears.

Within this pane, do the following:

1. Use the **Name**, **Style**, and **Size** controls to select a font. As you make selections from these controls, the *Sample text* in the center of the Font pane changes to reflect your current selections.
2. When you are satisfied with the selected font, click **Apply** to apply the font settings to the editor, then click **OK** to close the Application Preferences dialog.

RELATED CONCEPTS

“Editor Customization” on page 2

Customizing the Keyboard

You can customize the keyboard used to work in the editor with the **set keyAction** command.

The **set keyAction** command lets you program the logical keyboard and assign a command or set of commands to a key. This programming relates to a single file, so the same key can have a different effect for different files.

When a key is pressed, the editor performs an action dictated by the corresponding logical key. Many keys have default actions. Any key can be assigned an action. For example, the following command makes the Alt+A key scroll the text in a window up one screen:

```
set keyAction.a-a pageUp
```

Your operating system may reserve some keys, such as F1 and F10.

RELATED CONCEPTS

“Editor Customization” on page 2

RELATED REFERENCES

“keyAction parameter” on page 133

Changing text and background color palettes

You can select a different color palette for displaying text in the Editor pane. To do so, you must first open the Color palette pane in the Application Preferences dialog:

1. Select **File** from the editor menu bar.
2. Select **Preferences**. The Application Preferences dialog appear.
3. In the Application Preferences dialog, expand the **Editor** preferences tree by clicking on the + beside **Editor**.
4. Select **Color palette** from the expanded list of **Editor** preferences. The Color palette preference pane appears.

Within this pane, do the following:

1. Select a palette from the **Color palette** selection list.
2. Click **Apply** to apply the palette setting to the editor, then click **OK** to close the Application Preferences dialog.

RELATED CONCEPTS

“Editor Customization” on page 2

Chapter 6. Reference

Default editor commands

The editor window is fully programmable through the use of an extensive editor command and parameter set. You can use commands and parameters to customize the editor window, search for or change text in your document, or perform many other functions.

Select a command from the list below to display complete reference information for that command.

Command Name	Description
action	Used to run an editor command.
"add command" on page 26	Adds one or more blank lines into the current file.
"block command" on page 27	Sets or manipulates a block of text.
"compare command" on page 30	Examines two documents for differences in their content.
"delete command" on page 32	Deletes one or more lines from the current file.
"deleteText command" on page 33	Deletes text from the current file.
"expandAll command" on page 33	Brings hidden lines back into view.
"findText command" on page 34	Finds and optionally replaces text in a file.
"get command" on page 37	Gets a file and inserts it after the current line.
"help command" on page 38	Returns help information on a requested command or parameter.
"input command" on page 39	Gets user input to use with an editor command.
"insert command" on page 40	Inserts a new line into the current file.
"insertShow command" on page 40	Inserts a new <i>show</i> element into the current file.
"insertText command" on page 41	Inserts text at the current cursor position.
"load command" on page 42	Loads or reloads a file.
"locate command" on page 42	Moves the cursor to a specified element, line, mark, or sequence number.
"parse command" on page 44	Triggers the parser.
"print command" on page 44	Prints the current document.
"processPrefix command" on page 46	Processes prefix commands in the prefix area.
"query command" on page 47	Returns the setting of a parameter.
"replaceText command" on page 49	Replaces text at the current cursor position.
"resequence command" on page 50	Resequences sequence numbers in a file.
"save command" on page 50	Saves the contents of the current editing window into a file.
"screenShow command" on page 52	Forces all file views to be refreshed.
"set command" on page 52	Assigns a value to an editor parameter.
"undo command" on page 54	Undo one or more sets of file changes.
"updateProfile command" on page 55	Updates selected editor parameters as they apply to the current file.

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Editor parameters” on page 58

action command

The **action** command can be used to run an editor action.

Syntax

```
action actionName
```

Parameters

actionName

Use the *actionName* parameter to specify the name of the action that you want to run.

Status

The **status** parameter is not affected by this command.

Description

You may specify any of the default actions or user defined actions for *actionName*.

Examples

```
action undo  
action blockMarkLeft
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“actionClass parameter” on page 63 parameter

“actions parameter” on page 65 parameter

“updateProfile.userActions parameter” on page 211 parameter

“Default editor commands” on page 25

add command

The **add** command can be used to insert blank elements into a document.

Syntax

```
add [before] [show] [count]
```

Parameters

before

Use the optional parameter **before** to indicate that the new lines should be added before the current element.

`show`

Use the optional parameter **show** to indicate that the new elements should be **show** elements. Show elements are not saved with the document.

`count`

Must be a positive integer. The `count` parameter specifies the number of elements to be added. If you do not specify `count`, then 1 is assumed.

Status

The **status** parameter is not affected by this command.

Description

The **add** command can be used to insert one or more new elements into a document. It is similar to the **insert** and **insertShow** commands except that you cannot specify the element's text with the same command. You will need to use the **add** command to insert lines at the top of the file since the **insert** and **insertShow** commands add lines after the current line.

Examples

```
add 100
add before
add show
```

RELATED CONCEPTS

"Editor commands and parameters" on page 1

RELATED REFERENCES

"insert command" on page 40 command
"insertShow command" on page 40 command
"show parameter" on page 184 parameter
"text parameter" on page 191 parameter
"Default editor commands" on page 25

block command

The **block** command can be used to set the block selection or manipulate the text that is currently selected.

Syntax

```
block { clear
      | copy
      | delete
      | fill c
      | find [end]
      | lowerCase
      | move
      | overlay [transparent]
      | set [stream | character | element | rectangle]
      | shift [left | right] [count]
      | upperCase
    }
```

Parameters

You must specify at least one of the parameters to the block command.

clear	Use the clear parameter to remove the current block selection.
copy	Use the copy parameter to copy the currently selected text to the current cursor position. After the copy has completed, the new text will be selected.
delete	Use the delete parameter to delete the currently selected text from the document.
fill c	Use the fill parameter to replace all of the characters that are currently contained within the block selection with the character c.
find [end]	Use the find parameter to move the cursor to the beginning of the block selection. If you specify the optional end parameter, the cursor will be moved to the end of the block selection.
lowerCase	Use the lowerCase parameter to change the selected text to lower case.
move	Use the move parameter to move the selected text to the current cursor position. After the move has completed, the moved text will be reselected.
overlay [transparent]	Use the overlay parameter to overlay the text at the current cursor position with the currently selected text. If you specify the optional transparent parameter, then only the spaces will be overlaid by the selected text.
set [stream character element rectangle]	Use the set parameter to set the block selection. A new block selection will be created at the current cursor position if <ul style="list-style-type: none">• there is no block selection.• the block selection is not in the current view.• the specified block type does not match the current type.

If there is no block type specified and there is no block selection in the current view, then **current.block.defaultType** will be used. If there is a block selection in the current view and the specified block type matches the block type of the block selection or there is no specified block type, then the current block will be extended to the current cursor position.

shift [left | right] [*count*]

Use the **shift** parameter to shift the currently selected text. The **shift** parameter may only be used if the current block type is **element** or **rectangle**. Use the optional **left** and **right** parameters to indicate the direction that the text should be shifted. If you do not specify either, then the text will be shifted to the right. The *count* parameter may be specified to indicate the number of character positions that the text should be shifted. *count* must be an integer.

upperCase

Use the **upperCase** parameter to change the selected text to upper case.

Status

The **status** parameter is not affected by this command.

Description

There is only one block selection for the entire application. If you have a block selection in one document view and you set a block selection in a new document view then the block selection in the first document view is cleared. This allows you to perform copy and move operations from one document view to another. There is no ambiguity since there is only one selection.

There are four block selection types: stream, character, element, and rectangle.

- Stream selection allows you to select a stream of characters. The selection is coupled to the cursor. If you move the cursor, then the selection is removed. If you make a change while there is a stream selection active, then all of the selected text is removed before the change.
- Character selection also allows you to select a stream of characters but the selection is not coupled to the cursor.
- Rectangle selection allows you to select a rectangle of text. The selection is not coupled to the cursor.
- Element selection allows you to select whole elements only. The selection is not coupled to the cursor.

Block operations will only affect visible elements.

Examples

```
block clear  
block move  
block set
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“findText command” on page 34 command

“block.defaultType parameter” on page 67 parameter

blockCopy action

blockDelete action

blockFill action

blockLowerCase action

blockMarkBottom action

blockMarkCharacter action

blockMarkDown action

blockMarkElement action

blockMarkElementAtMouse action

blockMarkEnd action

blockMarkHome action

blockMarkLeft action

blockMarkNextWord action

blockMarkPageDown action

blockMarkPageLeft action

blockMarkPageRight action

blockMarkPageUp action

blockMarkPrevWord action

blockMarkRectangle action

blockMarkRectangleAtMouse action

blockMarkRight action

blockMarkToMouse action

blockMarkTop action

blockMarkUp action

blockMarkWord action

blockMarkWordAtMouse action

blockMove action

blockOverlay action

blockShiftLeft action

blockShiftRight action

blockUnmark action

blockUpperCase action

filterSelection action

findBlockEnd action

findBlockStart action

findSelection action

“Default editor commands” on page 25

compare command

The **compare** command can be used to compare the current document to a specified file. Lines that exist in the specified file but not in the current document are highlighted with the **styleAttributes.deletedLines** style attributes. Lines that exist in the current document but not in the specified file are highlighted with the **styleAttributes.addedLines** style attributes.

Syntax

```
compare { clear
          | next
          | previous
```

```

    refresh
    prompt [ "fileName" ]
    [ "fileName" ]
}

```

Parameters

clear	Use the parameter clear to indicate that you want to remove the compare information from a previous compare.
next	Use the parameter next to indicate that you want to move forward to the next mismatch. This will not wrap around to the top of the file.
previous	Use the parameter previous to indicate that you want to move backward to the previous mismatch. This will not wrap around to the bottom of the file.
refresh	Use the parameter refresh to refresh the comparison against the previously specified file, using the latest changes in the current view.
prompt ["fileName"]	Use the parameter prompt to indicate that the compare file dialog should be displayed to allow the user to select a file. If the optional parameter <i>fileName</i> is specified, the compare file dialog will be initialized with the specified file name.
["fileName"]	Use the parameter <i>fileName</i> to indicate the file that should be used. If no <i>filename</i> is specified, then the document will be compared with the saved version of the file.

Status

The **status** parameter will be set to one of the following:

null	Indicates that the compare command successfully located the specified file.
file.notFound	Indicates that the compare command could not locate the specified file.
file.errorReading	Indicates that the compare command encountered an error while reading the file.

Description

Lines which exist in the current document but not in the compare document are highlighted and continue to be part of the document. Lines that do not exist in the current document but exist in the compare document are inserted as **show** elements in the current document view. They are protected and cannot be edited. Since they are **show** elements, they will not be saved with the document. The comparison takes into account the settings of the **compare.parameter** parameters. After changing these settings another **compare** or **compare refresh** command must be issued before the view will reflect the changes.

Examples

```
compare prompt
compare "test.java"
compare clear
compare next
compare previous
compare refresh
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“compare.ignoreCase parameter” on page 74 parameter
“compare.ignoreLeadingBlanks parameter” on page 76 parameter
“compare.ignoreTrailingBlanks parameter” on page 78 parameter
“compare.ignoreAllBlanks parameter” on page 80 parameter
“show parameter” on page 184 parameter
“styleAttributes parameter” on page 187 parameter
compare action
compareClear action
compareNext action
comparePrevious action
compareRefresh action
“Default editor commands” on page 25

delete command

The **delete** command can be used to delete lines from a document.

Syntax

```
delete [ count ]
```

Parameters

count

Must be a positive integer. The *count* parameter specifies the number of lines that should be deleted. If you do not specify *count*, then 1 is assumed.

Status

The **status** parameter is not affected by this command.

Description

The **delete** command only affects visible elements.

Examples

```
delete
delete 10
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“add command” on page 26 command

“insert command” on page 40 command

“insertShow command” on page 40 command

“Default editor commands” on page 25

deleteText command

The **deleteText** command can be used to delete text from a document.

Syntax

```
deleteText [ count ]
```

Parameters

count

Must be a positive integer. The *count* parameter specifies the number of characters that should be deleted. If you do not specify *count*, then 1 is assumed.

Status

The **status** parameter is not affected by this command.

Description

The **deleteText** command will only delete characters in the current element.

Examples

```
deleteText  
deleteText 100
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“insertText command” on page 41 command

“replaceText command” on page 49 command

“text parameter” on page 191 parameter

“Default editor commands” on page 25

expandAll command

The **expandAll** command can be used to set the **expanded** parameter for every element in the document view. It also sets the **topExpanded** parameter for the current document view. If you do not specify any parameters, **on** is assumed.

Syntax

```
expandAll [ on | off ]
```

Parameters

on

Use the **on** parameter to set the **expanded** parameter to **on** for every element in the document view. This will also set the **topExpanded** parameter to **on**.

off

Use the **off** parameter to set the **expanded** parameter to **off** for every element in the document view. This will also set the **topExpanded** parameter to **off**.

Status

The **status** parameter is not affected by this command.

Description

When you have hidden elements in the document view, they can be forced visible by setting **expanded** or **topExpanded** to **on**. Issuing **expandAll on** will force all of the hidden elements in the document view to become visible. Issuing **expandAll off** will allow all of the hidden elements in the document view to remain hidden.

Examples

```
expandAll on  
expandAll off
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“findText command” command

“excludedClasses parameter” on page 89 parameter

“expandHide parameter” on page 90 parameter

“expanded parameter” on page 94 parameter

“forceAllVisible parameter” on page 120 parameter

“forceVisible parameter” on page 120 parameter

“includedClasses parameter” on page 127 parameter

“markExcluded parameter” on page 143 parameter

“markIncluded parameter” on page 147 parameter

“topExpanded parameter” on page 193 parameter

“visible parameter” on page 221 parameter

filterSelection action

“Default editor commands” on page 25

findText command

The **findText** command can be used to locate and optionally replace text in a document. The parameters for the **findText** command are divided into two groups. The first group of optional parameters includes **up**, **checkStart**, **replace**, and **all**. If only these options are specified then the **findText** command will use the values indicated by the **current.findText** parameters to perform the search. If you specify any of the parameters from the second group you must specify the *text* parameter. In this case, the **current.findText** parameters will be ignored and only the options specified will be taken into account.

Syntax

```
findText [ up
          | checkStart
          | replace
          | all
          ] [...]
          [ [ mark
            | columns col1 col2
            | block
            | noWrap
            | asis
            | noEmphasis
            | regularExpression
            | replaceWith replaceText
            | quiet
            ] [...]
          ]
          text
        ]
```

Parameters

up

Use the optional parameter **up** to indicate that the search should proceed backward from the current position.

checkStart

Use the optional parameter **checkStart** to indicate that the current cursor position should be checked before proceeding with the search.

replace

Use the optional parameter **replace** to indicate that the found text should be replaced by the text indicated by the **findText.replaceText** parameter.

all

Use the optional **all** parameter to indicate that all of the occurrences of the sought text should be located. The search ignores the current cursor position but it honors the **block** and **columns** restrictions.

mark

Use the optional **mark** parameter to indicate that the found text should be selected.

columns *col1 col2*

Use the optional parameter **columns** to restrict the scope of the search to the specified start and end columns. *col1* and *col2* must be positive integers.

block

Use the optional parameter **block** to restrict the scope of the search to the current block selection.

noWrap

Use the optional parameter **noWrap** to indicate that the search is to stop when it hits the bottom of the document. Or, in the case where the search is proceeding backward through the document, the **noWrap** parameter indicates that the search is to stop when the top of the document is reached.

asis

Use the optional parameter **asis** to indicate that the search should be a case sensitive search.

noEmphasis

Use the optional parameter **noEmphasis** to indicate that the found text is not to be emphasised with the document view's emphasis style.

<code>regularExpression</code>	Use the optional parameter regularExpression to indicate the the search string should be treated as a regular expression string.
<code>replaceWith <i>replaceText</i></code>	Use the optional parameter replaceWith to indicate that the found text should be replaced with <i>replaceText</i> . If <i>replaceText</i> contains spaces it must be quoted with quotes ("). If <i>replaceText</i> contains quotes, you must quote the quotes with a backslash (\"). If <i>replaceText</i> contains backslashes, you must quote the backslashes with a backslash (\\).
<code>quiet</code>	Use the optional parameter <code>quiet</code> to indicate that no messages or audio feedback should be issued.
<code>text</code>	Use the parameter <i>text</i> to indicate the text for which you wish to search. You may need to quote the string if you are searching for one of the findText keywords.

Status

The **status** parameter will be set to one of the following:

<code>null</code>	Indicates that the findText command successfully located the specified text without any special conditions.
<code>findText.onlyOccurrence</code>	Indicates that the findText command searched the entire document, wrapped and located the specified text at the original cursor location.
<code>findText.wrapped</code>	Indicates that the findText command successfully located the specified text but had to wrap around to the beginning of the file during the search. Or, if searching backward it had to wrap around to the end of the file during the search.
<code>findText.notFound</code>	Indicates that the specified text could not be found.

Description

The **findText** command uses the following parameters:

- `findText.asis`
- `findText.block`
- `findText.columns`
- `findText.emphasis`
- `findText.endColumn`
- `findText.findText`
- `findText.mark`
- `findText.regularExpression`
- `findText.replaceText`
- `findText.startColumn`
- `findText.wrap`

The **findText** command only affects visible elements.

Examples

```
findText "text"
findText replaceWith "new text" "text"
findText up
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“emphasisLength parameter” on page 88 parameter
“expandHide parameter” on page 90 parameter
“expanded parameter” on page 94 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“topExpanded parameter” on page 193 parameter
“visible parameter” on page 221 parameter
filterSelection action
find action
findAndReplace action
findAndReplaceNext action
findAndReplaceUp action
findNext action
findSelection action
findUp action
“Default editor commands” on page 25

get command

The **get** command can be used to import a file into the current document at the current cursor position.

Syntax

```
get { prompt [ "fileName" ]
      | "fileName"
    }
```

Parameters

prompt ["fileName"]

"fileName"

Use the parameter **prompt** to indicate that the get file dialog should be displayed to allow the user to select a file. If the optional parameter *fileName* is specified, the get file dialog will be initialized with the specified file name.

Use the parameter *fileName* to indicate the file that should be used.

Status

The **status** parameter will be set to one of the following:

null	Indicates that the get command successfully located and loaded the specified file.
file.notFound	Indicates that the get command could not locate the specified file.
findText.errorReading	Indicates that the get command encountered an error while reading the file.

Examples

```
get "test.java"  
get prompt
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

get action
“Default editor commands” on page 25

help command

The **help** command can be used to view help on the lpex commands and parameters.

Syntax

```
help [ command ]
```

Parameters

command

The *command* parameter may be used to specify the editor command for which help should be displayed. If you do not specify a *command*, then general command help is shown. *command* must match the editor command text exactly. If *command* includes parameters for an editor command, then help on the last parameter will be displayed.

Status

The **status** parameter is not affected by this command.

Description

The **help** command displays help for editor commands and their parameters. If help is not available or *command* is not recognized, then general help on the editor commands is displayed.

Examples

```
help  
help query  
help query hideSequenceNumbers
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“help.configuration parameter” on page 122 parameter

“help.location parameter” on page 123 parameter

“help.homepage parameter” on page 123 parameter

“Default editor commands” on page 25

input command

The **input** command can be used to get user input and then run an editor command.

Syntax

```
input [ "label" [ "text" ] ] "command"
```

Parameters

label

Use the *label* parameter to specify the prompt that you want to display for the user. If *label* contains quotes or backslashes, then they must be quoted with the backslash character. If you do not wish to specify a prompt string but you need to specify the *text* parameter then you must specify an empty string ("").

text

Use the *text* parameter to specify the text that you want to initialize the entry field with. If *text* contains quotes or backslashes, then they must be quoted with the backslash character.

command

Use the *command* parameter to specify the command that you want to run if the user hits enter from the input prompt. The command must include any parameters and trailing spaces. The text that the user entered into the entry field will be appended to the command's parameters. The command may be any valid editor command. If the command contains quotes or backslashes, then they must be quoted with the backslash character.

Status

The **status** parameter is not affected by this command.

Examples

```
input "Enter fill character:" "block fill "  
input "" "text" "set messageText "
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

blockFill action

locateLine action

nameMark action

rename action

“Default editor commands” on page 25

insert command

The **insert** command inserts a new line into a document.

Syntax

```
insert [ text ]
```

Parameters

text

The *text* parameter may be specified to indicate the text for the the new line.

Status

The **status** parameter is not affected by this command.

Description

The new element is inserted after the current element. If you need to insert an element at the top of the document you will need to use the **add** command. If you want to insert a **show** element use the **insertShow** command.

Examples

```
insert some text
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“add command” on page 26 command

“insertShow command” command

“text parameter” on page 191 parameter

“Default editor commands” on page 25

insertShow command

The **insertShow** command can be used to insert a new **show** element into a document. A **show** element is a line intended for display only and will not be saved when the document is saved.

Syntax

```
insertShow [ text ]
```

Parameters

text

The *text* parameter specifies the text for the new **show** element.

Status

The **status** parameter is not affected by this command.

Description

The new element is inserted after the current element. If you need to insert an element at the top of the document you will need to use the **add** command. If you want to insert a regular element (not a show element), then use the **insert** command.

Examples

```
insertShow This is a show element.
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“add command” on page 26 command

“insert command” on page 40 command

“text parameter” on page 191 parameter

“show parameter” on page 184 parameter

“Default editor commands” on page 25

insertText command

The **insertText** command can be used to insert text at the current cursor position.

Syntax

```
insertText text
```

Parameters

text

The text specified with the *text* parameter will be inserted into the document at the current cursor position.

Status

The **status** parameter is not affected by this command.

Description

If the text passed to the **insertText** command contains new line characters ('\n') then new lines will be inserted into the document.

Examples

```
insertText hello
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“deleteText command” on page 33 command

“replaceText command” on page 49 command

“text parameter” on page 191 parameter

“Default editor commands” on page 25

load command

The **load** command can be used to load (or reload) the file indicated by the current setting of the **name** parameter.

Syntax

```
load
```

Parameters

The **load** command has no parameters.

Status

The **status** parameter is not affected by this command.

Description

The **load** command will clear the undo stack, reset the changes count, and issue **updateProfile** for all of the views of the current document.

Examples

```
load
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“undo command” on page 54 command

“updateProfile command” on page 55 command

“changes parameter” on page 69 parameter

“dirty parameter” on page 85 parameter

“name parameter” on page 154 parameter

reload action

“Default editor commands” on page 25

locate command

The **locate** command can be used to move the cursor to the specified element, line, mark or sequence number.

Syntax

```
locate [ emphasis ] { element n
                      | line n
                      | mark {name | #id}
                      | sequenceNumber n
                      }
```

Parameters

emphasis

Use the **emphasis** parameter to indicate that the target should be emphasized if it is found.

element *n*

Use the **element** parameter to move the cursor to the specified element. *n* must be a positive integer. Document elements include both **show** lines and non **show** lines.

line *n*

Use the **line** parameter to move the cursor to the specified line. *n* must be a positive integer. Document lines do not include **show** lines.

mark {*name* | #*id*}

Use the **mark** parameter to move the cursor to the specified mark. A mark may be indicated by its name or by its id. *Id* must be a positive integer.

sequenceNumber *n*

Use the **sequenceNumber** parameter to move the cursor to the line with the specified sequence number. *n* must be a positive integer.

Status

The **status** parameter will be set to one of the following:

null

Indicates that the **locate** command successfully located the specified item.

locate.notFound

Indicates that the specified item could not be found.

Examples

```
locate element 100
locate mark a
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“element parameter” on page 86 parameter

“elements parameter” on page 88 parameter

“emphasisLength parameter” on page 88 parameter

“line parameter” on page 137 parameter

“lines parameter” on page 139 parameter

“mark parameter” on page 141 parameter

“markId parameter” on page 146 parameter

“sequenceNumber parameter” on page 180 parameter

“sequenceNumbers parameter” on page 182 parameter

findMark action

findQuickMark action

locateLine action

“Default editor commands” on page 25

parse command

The **parse** command can be used to trigger the parser.

Syntax

```
parse [ all ]
```

Parameters

all

Use the optional parameter **all** to indicate that the all of the lines in the document should be parsed. If this option is not specified, then only the lines that are currently on the parse pending list will be parsed.

Status

The **status** parameter is not affected by this command.

Examples

```
parse
parse all
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“parser parameter” on page 155 parameter

“Default editor commands” on page 25

print command

The **print** command can be used to print the current document.

Syntax

```
print [ block
      | bottomMargin n
      | endElement n
      | font fontName
      | leftMargin n
      | lineNumbers { on | off }
      | prompt
      | rightMargin n
      | startElement n
      | tokenized { on | off }
      | topMargin n
      | visible
      ] [...]
```

Parameters

block

Use the optional parameter **block** to indicate that only the selected text should be printed.

<code>bottomMargin <i>n</i></code>	Use the optional parameter bottomMargin to indicate the bottom margin. <i>n</i> must be an integer that is greater than or equal to zero. <i>n</i> indicates the number of pixels that should be used for the bottom margin. If you do not specify bottomMargin , then the setting is taken from the current.print.bottomMargin parameter.
<code>endElement <i>n</i></code>	Use the optional parameter endElement to indicate the last element that you want printed. <i>n</i> must be a positive integer. If you do not specify the end element, then printing continues to the end of the file.
<code>font <i>fontName</i></code>	Use the optional parameter font to indicate the name of the font that you want to use to print with. If you do not specify font , then the setting is taken from the current.print.font parameter. If the print.font parameter is not set, then the font is taken from the current view. The <i>fontName</i> parameter should be in one of the following forms: <ul style="list-style-type: none"> • fontname-style-pointsize • fontname-pointsize • fontname-style • fontname <p>where style is one of the three strings "bold", "bolditalic", or "italic", and pointsize is a decimal representation of the point size.</p>
<code>leftMargin <i>n</i></code>	Use the optional parameter leftMargin to indicate the left margin. <i>n</i> must be an integer that is greater than or equal to zero. <i>n</i> indicates the number of pixels that should be used for the left margin. If you do not specify leftMargin , then the setting is taken from the current.print.leftMargin parameter.
<code>lineNumbers { on off }</code>	Use the optional parameter lineNumbers to indicate if line numbers should be displayed to the left of each line. If you do not specify lineNumbers , then the setting is taken from the current.print.lineNumbers parameter.
<code>prompt</code>	Use the optional parameter prompt to indicate that the user should be prompted for print options.
<code>rightMargin <i>n</i></code>	Use the optional parameter rightMargin to indicate the right margin. <i>n</i> must be an integer that is greater than or equal to zero. <i>n</i> indicates the number of pixels that should be used for the right margin. If you do not specify rightMargin , then the setting is taken from the current.print.rightMargin parameter.
<code>startElement <i>n</i></code>	Use the optional parameter startElement to indicate the first element that you want printed. <i>n</i> must be a positive integer. If you do not specify a starting element, then printing starts at the top of the file.

tokenized { on | off }

Use the optional parameter **tokenized** to indicate if the text should be tokenized or not. If you do not specify **tokenized**, then the setting is taken from the **current.print.tokenized** parameter.

topMargin *n*

Use the optional parameter **topMargin** to indicate the top margin. *n* must be an integer that is greater than or equal to zero. *n* indicates the number of pixels that should be used for the top margin. If you do not specify **topMargin**, then the setting is taken from the **current.print.topMargin** parameter.

visible

Use the optional parameter **visible** to indicate that only visible lines should be printed.

Status

The **status** parameter is not affected by this command.

Examples

```
print
print tokenized
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“block command” on page 27 command
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
“visible parameter” on page 221 parameter
print action
“Default editor commands” on page 25

processPrefix command

The **processPrefix** command can be used to process the prefix commands in the prefix area.

Syntax

```
processPrefix profile
```

Parameters

profile

Use the *profile* parameter to indicate which profile should be used to process the prefix commands. The **processPrefix** command only processes the prefix commands if *profile* is **seu**, **xedit**, or **ispf**.

Status

The **status** parameter is not affected by this command.

Description

The **processPrefix** command will attempt to process all of the prefix commands in the prefix area starting at the top of the document and moving down until no more prefix commands remain. If a prefix command is encountered that is not recognized, it is left in the prefix area. Refer to the following profiles to see the prefix commands that are processed by the **processPrefix** command:

- ispf
- seu
- xedit

Examples

```
processPrefix seu
processPrefix xedit
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“baseProfile parameter” on page 66 parameter
“inPrefix parameter” on page 126 parameter
“lineNumbers parameter” on page 138 parameter
“prefixPosition parameter” on page 159 parameter
“prefixText parameter” on page 161 parameter
“updateProfile.baseProfile parameter” on page 194 parameter
processPrefix action
“Default editor commands” on page 25

query command

The **query** command can be used to query the following editor settings:

“actionClass parameter” on page 63	“actionKey parameter” on page 64	“actionKeyText parameter” on page 65
“actions parameter” on page 65	“baseProfile parameter” on page 66	“beep parameter” on page 66
“block.defaultType parameter” on page 67	“changes parameter” on page 69	“class parameter” on page 70
“classes parameter” on page 71	“commandClass parameter” on page 71	“commandLine parameter” on page 72
“commands parameter” on page 74	“compare.ignoreCase parameter” on page 74	“compare.ignoreLeadingBlanks parameter” on page 76
“compare.ignoreTrailingBlanks parameter” on page 78	“compare.ignoreAllBlanks parameter” on page 80	current
“cursorRow parameter” on page 82	“default parameter” on page 83	“dirty parameter” on page 85
“displayPosition parameter” on page 85	“documentId parameter” on page 86	“element parameter” on page 86
“elementClasses parameter” on page 87	“elements parameter” on page 88	“emphasisLength parameter” on page 88
“excludedClasses parameter” on page 89	“expandHide parameter” on page 90	“expandHideAreaWidth parameter” on page 92

“expandTabs parameter” on page 92	“expanded parameter” on page 94	“fields parameter” on page 95
“findText.asis parameter” on page 96	“findText.block parameter” on page 98	“findText.columns parameter” on page 100
“findText.emphasis parameter” on page 102	“findText.endColumn parameter” on page 104	“findText.findText parameter” on page 106
“findText.mark parameter” on page 108	“findText.regularExpression parameter” on page 110	“findText.replaceText parameter” on page 112
“findText.startColumn parameter” on page 114	“findText.wrap parameter” on page 116	“font parameter” on page 118
“forceAllVisible parameter” on page 120	“forceVisible parameter” on page 120	“headerMark parameter” on page 121
“help.configuration parameter” on page 122	“help.homepage parameter” on page 123	“help.location parameter” on page 123
“hex parameter” on page 124	“hideSequenceNumbers parameter” on page 125	“inPrefix parameter” on page 126
“includedClasses parameter” on page 127	“insertMode parameter” on page 128	“install parameter” on page 129
“installProfile parameter” on page 131	“keyAction parameter” on page 133	“keys parameter” on page 136
“length parameter” on page 137	“line parameter” on page 137	“lineNumbers parameter” on page 138
“lines parameter” on page 139	“maintainSequenceNumbers parameter” on page 140	“mark parameter” on page 141
“markExcluded parameter” on page 143	“markExcludedHeader parameter” on page 144	“markHighlight parameter” on page 145
“markId parameter” on page 146	“markIncluded parameter” on page 147	“markProtect parameter” on page 148
“markStyle parameter” on page 149	“messageLine parameter” on page 150	“messageText parameter” on page 151
“mouseAction parameter” on page 151	“mouseEvents parameter” on page 153	“name parameter” on page 154
“palette parameter” on page 154	“parser parameter” on page 155	“pixelPosition parameter” on page 155
“popup parameter” on page 156	“position parameter” on page 158	“prefixAreaWidth parameter” on page 158
“prefixPosition parameter” on page 159	“prefixProtect parameter” on page 160	“prefixText parameter” on page 161
“print.bottomMargin parameter” on page 162	“print.font parameter” on page 163	“print.leftMargin parameter” on page 165
“print.lineNumbers parameter” on page 167	“print.rightMargin parameter” on page 169	“print.tokenized parameter” on page 171
“print.topMargin parameter” on page 172	“readonly parameter” on page 174	“recording parameter” on page 175
“rowHeight parameter” on page 175	“rows parameter” on page 176	“save.textLimit parameter” on page 176
“save.trim parameter” on page 178	“scroll parameter” on page 179	“sequenceNumber parameter” on page 180
“sequenceNumberStyle parameter” on page 181	“sequenceNumbers parameter” on page 182	“show parameter” on page 184
“status parameter” on page 184	“statusLine parameter” on page 185	“style parameter” on page 186
“styleAttributes parameter” on page 187	“tabs parameter” on page 189	“text parameter” on page 191
“textAreaWidth parameter” on page 191	“textWidth parameter” on page 192	“topExpanded parameter” on page 193
“updateProfile.baseProfile parameter” on page 194	“updateProfile.extensions parameter” on page 196	“updateProfile.noParser parameter” on page 197
“updateProfile.palette parameter” on page 199	“updateProfile.paletteAttributes parameter” on page 201	“updateProfile.palettes parameter” on page 203
“updateProfile.parser parameter” on page 204	“updateProfile.parserAssociation parameter” on page 206	“updateProfile.parserClass parameter” on page 208

“updateProfile.parsers parameter” on page 210	“updateProfile.userActions parameter” on page 211	“updateProfile.userCommands parameter” on page 213
“updateProfile.userKeyActions parameter” on page 215	“updateProfile.userMouseActions parameter” on page 217	“updateProfile.userProfile parameter” on page 219
	“visible parameter” on page 221	

Description

The results of the query command are displayed on the message line.

Examples

```
query save.textLimit
query current.font
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Editor parameters” on page 58
“set command” on page 52 command
“Default editor commands” on page 25

replaceText command

The **replaceText** command can be used to replace text at the current cursor position.

Syntax

```
replaceText text
```

Parameters

text

The text specified with the *text* parameter will be used to replace text at the current cursor position.

Status

The **status** parameter is not affected by this command.

Description

If the text passed to the **replaceText** command contains new line characters ('\n') then new lines will be inserted into the document.

Examples

```
replaceText Some new text.
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“deleteText command” on page 33 command
“insertText command” on page 41 command
“text parameter” on page 191 parameter
“Default editor commands” on page 25

resequence command

The **resequence** command can be used to resequence the document's sequence numbers.

Syntax

```
resequence [ start [ increment ] ]
```

Parameters

start

Use the optional parameter *start* to indicate the sequence number that should be used for the first line in the document. If this parameter is not specified, then the **resequence** command will use 100 as the start value.

increment

Use the optional parameter *increment* to indicate the increment that should be used to resequence the sequence numbers. If this parameter is not specified, then the **resequence** command will use the *start* value as the increment value.

Status

The **status** parameter is not affected by this command.

Description

If you issue the **resequence** command, the document's sequence numbers will be reordered. The document will only have sequence numbers if the **sequenceNumbers** parameter has been set.

Examples

```
resequence
resequence 10 10
```

RELATED CONCEPTS

"Editor commands and parameters" on page 1

RELATED REFERENCES

"sequenceNumber parameter" on page 180 parameter
"sequenceNumbers parameter" on page 182 parameter
"Default editor commands" on page 25

save command

The **save** command can be used to save the current document.

Syntax

```
save [ prompt
      | visible
      | trim
```



```

| noTrim
| textLimit n
] [...]
[ "filename" ]

```

Parameters

prompt

Use the optional parameter **prompt** to prompt the user for the name of the file under which this document should be saved.

visible

Use the optional parameter **visible** to indicate that only the visible elements should be saved.

trim

Use the optional parameter **trim** to indicate that all of the document's lines should be trimmed of any trailing spaces. If you do not specify **trim** or **noTrim**, then the option is taken from the current setting of the **current.save.trim** parameter.

noTrim

Use the optional parameter **noTrim** to ensure that the document's lines are not trimmed. If you do not specify **trim** or **noTrim**, then the option is taken from the current setting of the **current.save.trim** parameter.

textLimit *n*

Use the optional parameter **textLimit** to indicate the maximum line length. *n* must be an integer that is greater than or equal to zero. If *n* is zero, then no maximum line length will be enforced. If there are lines in the document that are longer than *n*, then those lines will be truncated. If you do not specify **textLimit**, then the setting is taken from the **current.save.textLimit** parameter.

filename

The ***filename*** parameter may be used to specify the file name under which this document should be saved. If no filename is specified, then the document is saved under the name in the **name** setting.

Status

The **status** parameter will be set to one of the following:

null

Indicates that the save was successful.

save.failed

Indicates that the save was unsuccessful.

save.cancelled

Indicates that the save was cancelled by the user.

Examples

```

save
save "test.java"

```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“length parameter” on page 137 parameter
“name parameter” on page 154 parameter
“save.textLimit parameter” on page 176 parameter
“save.trim parameter” on page 178 parameter
“visible parameter” on page 221 parameter
save action
saveAs action
“Default editor commands” on page 25

screenShow command

The **screenShow** command can be used to force all of the document views to be refreshed.

Syntax

screenShow

Parameters

The **screenShow** command has no parameters.

Status

The **status** parameter is not affected by this command.

Description

You do not normally have to issue the **screenShow** command since it is done implicitly after every user action has completed. It may be necessary though if you are using the programming interface to work with a document view outside of the context of a user action.

Examples

screenShow

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

set command

The **set** command can be used to set the following editor settings:

“actionClass parameter” on page 63	“beep parameter” on page 66	“block.defaultType parameter” on page 67
“class parameter” on page 70	“commandClass parameter” on page 71	“commandLine parameter” on page 72

“compare.ignoreCase parameter” on page 74	“compare.ignoreLeadingBlanks parameter” on page 76	“compare.ignoreTrailingBlanks parameter” on page 78
“compare.ignoreAllBlanks parameter” on page 80	“cursorRow parameter” on page 82	“default parameter” on page 83
“displayPosition parameter” on page 85	“elementClasses parameter” on page 87	“emphasisLength parameter” on page 88
“excludedClasses parameter” on page 89	“expandHide parameter” on page 90	“expandTabs parameter” on page 92
“expanded parameter” on page 94	“fields parameter” on page 95	“findText.asis parameter” on page 96
“findText.block parameter” on page 98	“findText.columns parameter” on page 100	“findText.emphasis parameter” on page 102
“findText.endColumn parameter” on page 104	“findText.findText parameter” on page 106	“findText.mark parameter” on page 108
“findText.regularExpression parameter” on page 110	“findText.replaceText parameter” on page 112	“findText.startColumn parameter” on page 114
“findText.wrap parameter” on page 116	“font parameter” on page 118	“forceAllVisible parameter” on page 120
“forceVisible parameter” on page 120	“help.configuration parameter” on page 122	“help.homepage parameter” on page 123
“help.location parameter” on page 123	“hideSequenceNumbers parameter” on page 125	“inPrefix parameter” on page 126
“includedClasses parameter” on page 127	“insertMode parameter” on page 128	“installProfile parameter” on page 131
“keyAction parameter” on page 133	“lineNumbers parameter” on page 138	“maintainSequenceNumbers parameter” on page 140
“mark parameter” on page 141	“markExcluded parameter” on page 143	“markExcludedHeader parameter” on page 144
“markHighlight parameter” on page 145	“markIncluded parameter” on page 147	“markProtect parameter” on page 148
“markStyle parameter” on page 149	“messageLine parameter” on page 150	“messageText parameter” on page 151
“mouseAction parameter” on page 151	“name parameter” on page 154	“popup parameter” on page 156
“position parameter” on page 158	“prefixPosition parameter” on page 159	“prefixProtect parameter” on page 160
“prefixText parameter” on page 161	“print.bottomMargin parameter” on page 162	“print.font parameter” on page 163
“print.leftMargin parameter” on page 165	“print.lineNumbers parameter” on page 167	“print.rightMargin parameter” on page 169
“print.tokenized parameter” on page 171	“print.topMargin parameter” on page 172	“readonly parameter” on page 174
“recording parameter” on page 175	“save.textLimit parameter” on page 176	“save.trim parameter” on page 178
“scroll parameter” on page 179	“sequenceNumber parameter” on page 180	“sequenceNumberStyle parameter” on page 181
“sequenceNumbers parameter” on page 182	“status parameter” on page 184	“statusLine parameter” on page 185
“style parameter” on page 186	“styleAttributes parameter” on page 187	“tabs parameter” on page 189
“text parameter” on page 191	“topExpanded parameter” on page 193	“updateProfile.baseProfile parameter” on page 194
“updateProfile.noParser parameter” on page 197	“updateProfile.palette parameter” on page 199	“updateProfile.paletteAttributes parameter” on page 201
“updateProfile.parser parameter” on page 204	“updateProfile.parserAssociation parameter” on page 206	“updateProfile.parserClass parameter” on page 208
“updateProfile.userActions parameter” on page 211	“updateProfile.userCommands parameter” on page 213	“updateProfile.userKeyActions parameter” on page 215

Description

Not all of the parameters that are available with the **query** command can be set with the **set** command.

Examples

```
set insertMode off
set text Here is some text.
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“query command” on page 47 command

“Default editor commands” on page 25

undo command

The undo command can be used to undo or redo changes in the current document. It can also be used to explicitly close off the current change, clear all changes from the undo stack or reset the changes count to zero.

Syntax

```
undo [ n
      | check
      | clear
      | resetChanges
      ]
```

Parameters

<i>n</i>	Use the <i>n</i> parameter to indicate the number of changes that should be undone. If the <i>n</i> is negative, then $ n $ changes will be redone.
check	Use the check parameter to close off the current change.
clear	Use the clear parameter to remove all of the changes from the undo stack.
resetChanges	Use the resetChanges parameter to reset the changes count to zero.

Status

The **status** parameter is not affected by this command.

Description

If no parameters are specified, then the **undo** command will undo one change.

The **dirty** parameter may be used to determine if there is an incomplete change. If there is no incomplete change then **undo check** will not do anything. If there is an incomplete change, then undo check will complete the change, increment the **changes** count and reset the **dirty** parameter to **off**.

Only complete changes can be undone.

Changes are implicitly completed when the cursor is moved off the current line.

Changes made while **recording** is **off** cannot be undone.

Changes made which only affect **show** elements are not recorded and cannot be undone.

Examples

```
undo
undo -1
undo clear
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“load command” on page 42 command

“save command” on page 50 command

“changes parameter” on page 69 parameter

“dirty parameter” on page 85 parameter

“recording parameter” on page 175 parameter

redo action

undo action

“Default editor commands” on page 25

updateProfile command

The **updateProfile** command can be used to update the current document view’s profile.

Syntax

```
updateProfile [ all ]
```

Parameters

all

Use the optional parameter **all** to indicate that you want to update the profiles of all of the document views that are currently open. Note that if **updateProfile** has never been issued for a document view, then issuing **updateProfile all** will not affect that document view.

Status

The **status** parameter is not affected by this command.

Description

The **updateProfile** command is normally issued when a document view is created but it may be issued at any time to allow the document view to reflect changes to the profile.

The following is the sequence of events that occur when the **updateProfile** command is issued against a document view:

1. The **readonly** parameter is set to **off**.
2. The **includedClasses** parameter is set such that all classes are included.
3. The **excludedClasses** parameter is set such that no classes are excluded.
4. Any registered **classes** are deregistered.
5. The style characters set by the **styleAttributes** parameter are cleared.
6. The **popup** parameter is reset to the **default**.
7. The **expandTabs** parameter is reset to **default**.
8. The **hideSequenceNumbers** parameter is reset to **default**.
9. The **sequenceNumberStyle** parameter is reset to **'!'**.
10. The **fields** parameter is reset such that there are no editing fields.
11. The **tabs** parameter is reset to **default**.
12. All of the **show** elements for this view are deleted.
13. The **style** parameter for the current view's view of all of the document's elements is reset.
14. The **elementClasses** parameter for the current view's view of all of the document's elements is reset.
15. The base profile is set to the value returned by **current.updateProfile.baseProfile**.
16. If **baseProfile** is **ispf**, **seu**, or **xedit** then **prefixProtect** is set to **off**. Otherwise, **prefixProtect** is set to **on**.
17. If **baseProfile** is **ispf**, **seu**, or **xedit** then **lineNumbers** is set to **on**. Otherwise, **lineNumbers** is set to **default**.
18. The color palette is set to the value returned by **current.updateProfile.palette**.
19. The built in styles are assigned style attributes based on the values returned by **current.updateProfile.paletteAttributes.style.palette** where *style* is substituted for each of the built in styles and *palette* is the value returned by the **palette** parameter. Refer to the **styleAttributes** parameter for information on the built in styles.
20. All of the **keys** settings are cleared.
21. All of the **mouseEvents** settings are cleared.
22. All of the **actions** settings are cleared.
23. New **keys** settings based on **baseProfile** are set.
24. New **mouseEvents** settings based on **baseProfile** are set.
25. New **actions** settings based on **current.updateProfile.userActions** are set.
26. Additional **keys** settings based on **current.updateProfile.userKeyActions** are set.
27. Additional **mouseEvents** settings base on **current.updateProfile.userMouseActions** are set.
28. All of the **commands** settings are cleared.
29. New **commands** settings based on **current.updateProfile.userCommands** are set.
30. The user profile from **current.updateProfile.userProfile** is issued.
31. If the current view has a parser, then the **resetParser** method of **LpexParser** is issued and the parser is discarded.
32. The parse pending list is cleared.

33. If **current.updateProfile.noParser** is set to **off** then the new parser name is retrieved from **current.updateProfile.parser**.
34. If the parser name is **associated**, then the parser name is retrieved from the **current.updateProfile.parserAssociation.extension** parameter.
35. The parser class is retrieved from **current.updateProfile.parserClass.parserName**.
36. An instance of the parser is created.
37. The **totalParse** method of **LpexParser** is issued.
38. If there are any **LpexViewListener** objects listening to this view, then the **updateProfile** method is issued.
39. If the document has a **name**, then the readonly attribute of the file is queried. If the file is readonly, then the **readonly** parameter is set to **on**.

The **updateProfile** command must be issued after any change to one or more of the **updateProfile.parameter** parameters in order to have the change reflected in any of the active document views.

Note that if you are constructing your own **LpexView** object and you customize it by adding actions, commands, key settings or mouse event settings, the **updateProfile** command will eliminate these settings unless you perform customizations of this sort within the **updateProfile** method of an **LpexViewListener**.

Examples

```
updateProfile  
updateProfile all
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“load command” on page 42 command
“parse command” on page 44 command
“actionClass parameter” on page 63 parameter
“actions parameter” on page 65 parameter
“baseProfile parameter” on page 66 parameter
“class parameter” on page 70 parameter
“classes parameter” on page 71 parameter
“commandClass parameter” on page 71 parameter
“commands parameter” on page 74 parameter
“elementClasses parameter” on page 87 parameter
“expandTabs parameter” on page 92 parameter
“fields parameter” on page 95 parameter
“hideSequenceNumbers parameter” on page 125 parameter
“keyAction parameter” on page 133 parameter
“keys parameter” on page 136 parameter
“lineNumbers parameter” on page 138 parameter
“mouseAction parameter” on page 151 parameter
“mouseEvents parameter” on page 153 parameter
“name parameter” on page 154 parameter
“palette parameter” on page 154 parameter
“parser parameter” on page 155 parameter
“popup parameter” on page 156 parameter
“prefixProtect parameter” on page 160 parameter
“readonly parameter” on page 174 parameter
“show parameter” on page 184 parameter
“style parameter” on page 186 parameter
“styleAttributes parameter” on page 187 parameter
“tabs parameter” on page 189 parameter
“updateProfile.baseProfile parameter” on page 194 parameter
“updateProfile.extensions parameter” on page 196 parameter
“updateProfile.noParser parameter” on page 197 parameter
“updateProfile.palette parameter” on page 199 parameter
“updateProfile.paletteAttributes parameter” on page 201 parameter
“updateProfile.palettes parameter” on page 203 parameter
“updateProfile.parser parameter” on page 204 parameter
“updateProfile.parserAssociation parameter” on page 206 parameter
“updateProfile.parserClass parameter” on page 208 parameter
“updateProfile.parsers parameter” on page 210 parameter
“updateProfile.userActions parameter” on page 211 parameter
“updateProfile.userCommands parameter” on page 213 parameter
“updateProfile.userKeyActions parameter” on page 215 parameter
“updateProfile.userMouseActions parameter” on page 217 parameter
“updateProfile.userProfile parameter” on page 219 parameter
setParser action
“Default editor commands” on page 25

Editor parameters

The editor window is fully programmable through the use of an extensive editor command and parameter set. You can use commands and parameters to customize the editor window, search for or change text in your document, or perform many other functions.

Select a parameter from the list below to display complete reference information for that parameter. Parameters can be set or queried unless noted otherwise.

Parameter Name	Description
"actionClass parameter" on page 63	Defines a user defined action for the current view.
"actionKey parameter" on page 64	Returns the primary key that is assigned to a specified action. Query only.
"actionKeyText parameter" on page 65	Obtains a translated text string for the primary key that is assigned to the specified action.
"actions parameter" on page 65	Lists all actions defined for the current view. Query only.
"baseProfile parameter" on page 66	Returns the base profile that was used the last time the updateProfile command was issued. Query only.
"beep parameter" on page 66	Determines if a beep is emitted when the screenShow command is issued.
"block.defaultType parameter" on page 67	Sets or queries the default block type
"changes parameter" on page 69	Returns the number of completed undoable changes made since the last save. Query only.
"class parameter" on page 70	Registers or deregister classes for the current view.
"classes parameter" on page 71	Returns a list of all of the classes that are currently registered for the current view. Query only.
"commandClass parameter" on page 71	Defines a user defined command for the current view.
"commandLine parameter" on page 72	Queries or sets the visibility of the command line.
"commands parameter" on page 74	Returns a list all of the commands defined for the current view. Query only.
"compare.ignoreCase parameter" on page 74	Determines whether differences in character case are significant when comparing files with the compare command.
"compare.ignoreLeadingBlanks parameter" on page 76	Determines whether differences in the number of blanks at the start of a line are significant when comparing files with the compare command.
"compare.ignoreTrailingBlanks parameter" on page 78	Determines whether differences in the number of blanks at the end of a line are significant when comparing files with the compare command.
"compare.ignoreAllBlanks parameter" on page 80	Determines whether differences in the number of blanks in a line are significant when comparing files with the compare command.
current	Returns the current setting of a parameter. Query only.
"cursorRow parameter" on page 82	Sets or queries the physical row of the line containing the cursor.
"default parameter" on page 83	Sets the default setting of a parameter.
"dirty parameter" on page 85	Indicates if there are unsaved changes in the current file. Query only.
"displayPosition parameter" on page 85	Sets or queries the cursor's display column position in the current element.
"documentId parameter" on page 86	Returns a unique integer that can be used to identify the document. Query only.
"element parameter" on page 86	Returns the ordinal number of the current element. Query only.
"elementClasses parameter" on page 87	Sets or queries classes for the current element.
"elements parameter" on page 88	Returns the total number of elements in the current document. Query only.
"emphasisLength parameter" on page 88	Sets or queries the number of characters that are emphasized.
"excludedClasses parameter" on page 89	Sets or queries classes that should be excluded from the current view.
"expandHide parameter" on page 90	Set or queries the visibility of the expand/hide area.

Parameter Name	Description
"expandHideAreaWidth parameter" on page 92	Returns the width in pixels of the expand/hide area. Query only.
"expandTabs parameter" on page 92	Sets or queries the state of tab character expansion.
"expanded parameter" on page 94	Sets or queries the visibility of hidden elements between the current element and the next visible element.
"fields parameter" on page 95	Sets or queries the current editing fields.
"findText.asis parameter" on page 96	Sets or queries case sensitivity setting for the findText command.
"findText.block parameter" on page 98	Sets or queries if find operations are restricted to selected text.
"findText.columns parameter" on page 100	Sets or queries if find operations are restricted to selected columns of text.
"findText.emphasis parameter" on page 102	Sets or queries whether found text is emphasized.
"findText.endColumn parameter" on page 104	Sets or queries the end column used if find operations are restricted to a column range.
"findText.findText parameter" on page 106	Sets or queries text to be found by the findText command.
"findText.mark parameter" on page 108	Sets or queries whether found text is selected.
"findText.regularExpression parameter" on page 110	Sets or queries the regular expression setting for the findText command.
"findText.replaceText parameter" on page 112	Sets or queries replacement text used by the findText command.
"findText.startColumn parameter" on page 114	Sets or queries the start column used if find operations are restricted to a column range.
"findText.wrap parameter" on page 116	Sets or queries the wrap setting for the findText command.
"font parameter" on page 118	Sets or queries the current font used to display text in the edit window.
"forceAllVisible parameter" on page 120	Sets or queries forced visibility of all of document elements.
"forceVisible parameter" on page 120	Sets or queries forced visibility of the current element.
"headerMark parameter" on page 121	Determine if the current element is a header element for an excluded mark. (Query only)
"help.configuration parameter" on page 122	Sets or queries the configuration file used by the help command.
"help.homepage parameter" on page 123	Sets or queries the default help page that is displayed when no other help is available for a help request.
"help.location parameter" on page 123	Sets or queries the location of properties files used to map help requests to the appropriate html pages.
"hex parameter" on page 124	Returns the hexadecimal value of the character at the current cursor location. Query only.
"hideSequenceNumbers parameter" on page 125	Sets or queries whether document sequence numbers should be visible or not.
"inPrefix parameter" on page 126	Sets or queries the cursor location in the prefix or text area.
"includedClasses parameter" on page 127	Sets or queries classes that should be included in the current view.
"insertMode parameter" on page 128	Sets or queries the current insertion mode.
"install parameter" on page 129	Returns the installation setting of a parameter. Query only.
"installProfile parameter" on page 131	Sets or queries the name of a java properties file used to specify installation settings for the editor.
"keyAction parameter" on page 133	Sets or queries the action assignment for a specified key.
"keys parameter" on page 136	Lists all keys to which actions have been assigned. Query only.
"length parameter" on page 137	Returns the length of the current element. Query only.

Parameter Name	Description
“line parameter” on page 137	Returns the line number of the current element. Query only.
“lineNumbers parameter” on page 138	Sets or queries visibility of the line numbers.
“lines parameter” on page 139	Returns the total number of lines in the current document. Query only.
“maintainSequenceNumbers parameter” on page 140	Sets or queries whether the editor is maintaining the current document’s sequence numbers.
“mark parameter” on page 141	Sets, queries or removes a named mark.
“markExcluded parameter” on page 143	Sets or queries the excluded attribute of a mark.
“markExcludedHeader parameter” on page 144	Sets or queries whether a specified mark should have a header element when it is excluded.
“markHighlight parameter” on page 145	Sets or queries whether a specified mark should be highlighted.
“markId parameter” on page 146	Returns the id number of a specified mark. Query only.
“markIncluded parameter” on page 147	Sets or queries the included attribute of a mark.
“markProtect parameter” on page 148	Sets or queries the protect attribute of a mark.
“markStyle parameter” on page 149	Sets or queries the style character associated with a mark.
“messageLine parameter” on page 150	Sets or queries the visibility of the message line.
“messageText parameter” on page 151	Sets or queries text displayed in the message line.
“mouseAction parameter” on page 151	Sets or queries the action assignment for a specified mouse event.
“mouseEvents parameter” on page 153	Lists all mouse events to which actions have been assigned. Query only.
“name parameter” on page 154	Sets or queries the current document’s file name.
“palette parameter” on page 154	Returns the color palette in use the last time the updateProfile command was issued. Query only.
“parser parameter” on page 155	Returns the name of the parser used by the current view. Query only.
“pixelPosition parameter” on page 155	Returns the pixel offset of the current cursor position. Query only.
“popup parameter” on page 156	Sets or queries the contents of the pop-up menu.
“position parameter” on page 158	Sets or queries the cursor column position on the current element.
“prefixAreaWidth parameter” on page 158	Returns the width in pixels of the prefix area. Query only.
“prefixPosition parameter” on page 159	Sets or queries the cursor column position in the prefix area.
“prefixProtect parameter” on page 160	Sets or queries if the the cursor can be moved into the prefix area.
“prefixText parameter” on page 161	Sets or queries the current element’s prefix text.
“print.bottomMargin parameter” on page 162	Sets or queries the bottom margin used by the print command.
“print.font parameter” on page 163	Sets or queries the font used by the print command.
“print.leftMargin parameter” on page 165	Sets or queries the left margin used by the print command.
“print.lineNumbers parameter” on page 167	Sets or queries if the print command prints line numbers at the start of each line.
“print.rightMargin parameter” on page 169	Sets or queries the right margin used by the print command.
“print.tokenized parameter” on page 171	Sets or queries if the print command prints tokenized attributes displayed by the parser.
“print.topMargin parameter” on page 172	Sets or queries the top margin used by the print command.
“readonly parameter” on page 174	Sets or queries if the document can be edited with the current view.

Parameter Name	Description
“recording parameter” on page 175	Sets or queries if the editor is recording changes to the file.
“rowHeight parameter” on page 175	Returns the height in pixels of the current row of text. Query only.
“rows parameter” on page 176	Returns the number of rows of text that can be displayed in the text window. Query only.
“save.textLimit parameter” on page 176	Sets or queries the maximum line length used by the save command.
“save.trim parameter” on page 178	Sets or queries if the save command should trim trailing blanks from all of the lines in the document.
“scroll parameter” on page 179	Sets or queries the number of pixels that the current view is scrolled to the right.
“sequenceNumber parameter” on page 180	Sets or queries the current element’s sequence number.
“sequenceNumberStyle parameter” on page 181	Sets or queries the style character that is used to display the sequence numbers in the current view.
“sequenceNumbers parameter” on page 182	Sets or queries the starting column and width of the sequence numbers.
“show parameter” on page 184	Determine if the current view’s current element is a show element.
“status parameter” on page 184	Sets or queries the command status.
“statusLine parameter” on page 185	Sets or queries the visibility of the status line.
“style parameter” on page 186	Sets or queries style characters that used to display the text on the current element..
“styleAttributes parameter” on page 187	Sets or queries style attributes for a style character or for one of the built in styles.
“tabs parameter” on page 189	Set or queries tab stops used by the nextTabStop and prevTabStop actions.
“text parameter” on page 191	Sets or queries the current element’s text.
“textAreaWidth parameter” on page 191	Returns the width in pixels of the text area. Query only.
“textWidth parameter” on page 192	Returns the width in pixels of the current element’s text.
“topExpanded parameter” on page 193	Sets or queries the visibility of hidden elements between the top of the document and the first visible element.
“updateProfile.baseProfile parameter” on page 194	Sets or queries the base profile that will be used by the updateProfile command.
“updateProfile.extensions parameter” on page 196	Lists all of file extensions that are associated with parsers. Query only.
“updateProfile.noParser parameter” on page 197	Sets or queries if the updateProfile command should set a parser.
“updateProfile.palette parameter” on page 199	Sets or queries the color palette that will be used by the updateProfile command.
“updateProfile.paletteAttributes parameter” on page 201	Sets or queries style attributes used for a specified style and palette when the updateProfile command is issued.
“updateProfile.palettes parameter” on page 203	Lists all color palettes available to the updateProfile command. Query only.
“updateProfile.parser parameter” on page 204	Sets or queries the parser used by the updateProfile command.
“updateProfile.parserAssociation parameter” on page 206	Sets or queries the parser associated with a file extension.
“updateProfile.parserClass parameter” on page 208	Sets or queries the class name for a parser.
“updateProfile.parsers parameter” on page 210	Lists all parsers that are available to the updateProfile command. Query only.
“updateProfile.userActions parameter” on page 211	Sets or queries user actions used by the updateProfile command.
“updateProfile.userCommands parameter” on page 213	Sets or queries user commands used by the updateProfile command.
“updateProfile.userKeyActions parameter” on page 215	Sets or queries user key actions used by the updateProfile command.

Parameter Name	Description
“updateProfile.userMouseActions parameter” on page 217	Sets or queries user mouse actions used by the updateProfile command.
“updateProfile.userProfile parameter” on page 219	Sets or queries the user profile used by the updateProfile command.
“visible parameter” on page 221	Determine if the current view’s current element is visible in the current view.

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“query command” on page 47

“set command” on page 52

“Default editor commands” on page 25

actionClass parameter

The **actionClass** parameter can be used to define a user defined action for the current view. You can also use the **actionClass** parameter to redefine the default behaviour for one of the default actions.

Availability

query command
set command

Scope

The current view.

Syntax

```
query actionClass.action
set actionClass.action [className]
```

Parameters

action

Use the *action* qualifier to specify the name of the editor action that you wish to set or query. If you are using the **set** command, you may specify a default editor action in order to override its default behaviour.

className

Use the *className* parameter to specify the name of a class that implements the `LpexAction` interface. The class must have a default constructor. If you do not specify the *className* parameter, then the specified action will be reset to it’s installation setting.

Examples

```
query actionClass.test
set actionClass.test com.ibm.lpex.samples.TestAction
set actionClass.test
```

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“actions parameter” on page 65 parameter
“keyAction parameter” on page 133 parameter
“mouseAction parameter” on page 151 parameter
“updateProfile.userActions parameter” on page 211 parameter
“updateProfile.userKeyActions parameter” on page 215 parameter
“updateProfile.userMouseActions parameter” on page 217 parameter
“Default editor commands” on page 25
“Default editor actions” on page 222

actionKey parameter

Returns the primary key that is assigned to a specified action.

Availability

query command

Scope

The current view.

Syntax

```
query actionKey.action
```

Parameters

action

Use the *action* qualifier to specify the name of the editor action for which you wish to determine its primary key setting.

Examples

```
query actionKey.find
```

RELATED REFERENCES

“query command” on page 47 command
“actionKeyText parameter” on page 65 parameter
“keyAction parameter” on page 133 parameter

actionKeyText parameter

The **actionKeyText** parameter can be used to obtain a translated text string for the primary key that is assigned to the specified action.

Availability

query command

Scope

The current view.

Syntax

```
query actionKeyText.action
```

Parameters

action

Use the *action* qualifier to specify the name of the editor action for which you wish to obtain a translated text string for its primary key setting.

Examples

```
query actionKeyText.find
```

RELATED REFERENCES

“query command” on page 47 command
“actionKey parameter” on page 64 parameter
“keyAction parameter” on page 133 parameter

actions parameter

The **actions** parameter can be used to list all of the actions that are defined for the current view. Note that this list does not include the default actions unless they have been redefined.

Availability

query command

Scope

The current view.

Syntax

```
query actions
```

Examples

```
query actions
```

RELATED REFERENCES

“query command” on page 47 command
“actionClass parameter” on page 63 parameter

baseProfile parameter

The **baseProfile** parameter can be used to determine the base editor profile that was used the last time the **updateProfile** command was issued. The **updateProfile** command is normally issued as part of opening a file. The base editor profile is used to determine the basic look and feel of the editor. Java lpex currently recognizes the following base profiles: **brief**, **epm**, **ispf**, **lpex**, **seu**, and **xedit**.

Availability

query command

Scope

The current view.

Syntax

```
query baseProfile
```

Examples

```
query baseProfile
```

RELATED CONCEPTS

“Using alternate editor profiles” on page 2

RELATED REFERENCES

“query command” on page 47 command

“updateProfile command” on page 55 command

“updateProfile.baseProfile parameter” on page 194 parameter

“brief base profile” on page 231

“epm base profile” on page 238

“ispf base profile” on page 244

“lpex base profile” on page 254

“seu base profile” on page 260

“xedit base profile” on page 271

beep parameter

The **beep** parameter may be used to query or set if a beep should be emitted the next time the **screenShow** command is issued.

Availability

query command

set command

Scope

Global.

Syntax

```
query beep
set beep { on | off }
```

Parameters

on

Use the **on** parameter to indicate that a beep should be emitted the next time the **screenShow** command is issued.

off

Use the **off** parameter to indicate that a beep should not be emitted the next time the **screenShow** command is issued.

Examples

```
query beep
set beep on
```

RELATED REFERENCES

“query command” on page 47 command

“screenShow command” on page 52

command

“set command” on page 52 command

block.defaultType parameter

The **block.defaultType** parameter may be used to query or set the default block type. The default block type is used when no block type is specified. Many of the default editor actions use the default block type to determine how they should behave.

Availability

query command

set command

current parameter

default parameter

install parameter

Scope

block.defaultType is scoped to the current view.

current.block.defaultType is scoped to the current view.

default.block.defaultType is globally scoped.

install.block.defaultType is globally scoped.

Syntax

```
query block.defaultType
set block.defaultType [ default | stream | character | rectangle | element ]
query current.block.defaultType
query default.block.defaultType
set default.block.defaultType [ install | stream | character | rectangle | element ]
query install.block.defaultType
```

Parameters

default	If you specify the default parameter for the set block.defaultType command, then the current view will use the value of default.block.defaultType for the default block type.
stream	If you specify the stream parameter for the set block.defaultType command, then the current view will use stream as the default block type. If you specify the stream parameter for the set default.block.defaultType command, then all of the document views that have block.defaultType set to default will use stream as the default block type.
character	If you specify the character parameter for the set block.defaultType command, then the current view will use character as the default block type. If you specify the character parameter for the set default.block.defaultType command, then all of the document views that have block.defaultType set to default will use character as the default block type.
rectangle	If you specify the rectangle parameter for the set block.defaultType command, then the current view will use rectangle as the default block type. If you specify the rectangle parameter for the set default.block.defaultType command, then all of the document views that have block.defaultType set to default will use rectangle as the default block type.
element	If you specify the element parameter for the set block.defaultType command, then the current view will use element as the default block type. If you specify the element parameter for the set default.block.defaultType command, then all of the document views that have block.defaultType set to default will use element as the default block type.
install	If you specify the install parameter for the set default.block.defaultType command, then all of the views that have block.defaultType set to default will use the value of install.block.defaultType for the default block type.

Description

If you do not specify any of the parameters for the **set block.defaultType** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.block.defaultType** command, then **install** is assumed.

The **query current.block.defaultType** command will return the default block type for the current view. That is, it will return one of **stream**, **character**, **rectangle**, or **element**.

Examples

```
query block.defaultType
set block.defaultType character
query current.block.defaultType
query default.block.defaultType
set default.block.defaultType rectangle
query install.block.defaultType
```

RELATED REFERENCES

“block command” on page 27 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
blockMarkBottom action
blockMarkDown action
blockMarkEnd action
blockMarkHome action
blockMarkLeft action
blockMarkNextWord action
blockMarkPageDown action
blockMarkPageLeft action
blockMarkPageRight action
blockMarkPageUp action
blockMarkPrevWord action
blockMarkRight action
blockMarkToMouse action
blockMarkTop action
blockMarkUp action
blockMarkWord action
blockMarkWordAtMouse action

changes parameter

The **changes** parameter can be used to determine the number of completed undoable changes that have been made since the last save. If the current change has not been completed, then the current change will not be reflected by the **changes** parameter. You can use the **dirty** parameter to determine if there is an uncompleted change.

Availability

query command

Scope

The current view.

Syntax

```
query changes
```

Examples

query changes

RELATED REFERENCES

“query command” on page 47 command

“save command” on page 50 command

“undo command” on page 54 command

“dirty parameter” on page 85 parameter

class parameter

The **class** parameter may be used to register or deregister classes for the current view. Once registered, classes can be used to categorize the document’s elements. Most parsers assign classes to document elements. Classes are assigned to elements with the **elementClasses** parameter.

Availability

query command

set command

Scope

The current view.

Syntax

```
query class.className  
set class.className { on | off }
```

Parameters

className

Use the *className* qualifier to indicate the name of the class that you want to register or deregister.

on

Specify **on** if you want to register the specified class.

off

Specify **off** if you want to deregister the specified class.

Examples

```
query class.Message  
set class.Message on
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“classes parameter” on page 71 parameter

“elementClasses parameter” on page 87 parameter

“excludedClasses parameter” on page 89 parameter

“includedClasses parameter” on page 127 parameter

classes parameter

The **classes** parameter can be used to obtain a list of all of the classes that are currently registered for the current view. Classes are registered with the **class** parameter.

Availability

query command

Scope

The current view.

Syntax

`query classes`

Examples

`query classes`

RELATED REFERENCES

“query command” on page 47 command

“class parameter” on page 70 parameter

“elementClasses parameter” on page 87 parameter

“excludedClasses parameter” on page 89 parameter

“includedClasses parameter” on page 127 parameter

commandClass parameter

The **commandClass** parameter can be used to define a user defined command for the current view. You can also use the **commandClass** parameter to redefine the default behaviour for one of the default editor commands.

Availability

query command

set command

Scope

The current view.

Syntax

`query commandClass.command`
`set commandClass.command [className]`

Parameters

command

Use the *command* qualifier to specify the name of the editor command that you wish to set or query. If you are using the **set** command, you may specify a default editor command in order to override its default behaviour.

className

Use the *className* parameter to specify the name of a class that implements the **com.ibm.lplex.core.LplexCommand** interface. The class must have a default constructor. If you do not specify the *className* parameter, then the specified command will be reset to its installation setting.

Examples

```
query commandClass.test
set commandClass.test com.ibm.lplex.samples.TestCommand
set commandClass.test
```

RELATED REFERENCES

“Default editor commands” on page 25
“query command” on page 47 command
“set command” on page 52 command
“commands parameter” on page 74
parameter

commandLine parameter

The **commandLine** parameter may be used to query or set the visibility of the command line.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

commandLine is scoped to the current view.
current.commandLine is scoped to the current view.
default.commandLine is globally scoped.
install.commandLine is globally scoped.

Syntax

```
query commandLine
set commandLine [ default | on | off ]
query current.commandLine
query default.commandLine
set default.commandLine [ install | on | off ]
query install.commandLine
```

Parameters

default

If you specify the **default** parameter for the **set commandLine** command, then the current view will use the value of **default.commandLine** to determine the visibility of the command line.

on

If you specify the **on** parameter for the **set commandLine** command, then the command line will be visible on the current view. If you specify the **on** parameter for the **set default.commandLine** command, then the command line will be visible for all views that have **commandLine** set to **default**.

off

If you specify the **off** parameter for the **set commandLine** command, then the command line will not be visible on the current view. If you specify the **off** parameter for the **set default.commandLine** command, then the command line will not be visible for all views that have **commandLine** set to **default**.

install

If you specify the **install** parameter for the **set default.commandLine** command, then all of the views that have **commandLine** set to **default** will use the value of **install.commandLine** to determine the visibility of the command line.

Description

If you do not specify any of the parameters for the **set commandLine** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.commandLine** command, then **install** is assumed.

The **query current.commandLine** command will return **on** if the command line is visible or **off** if the command line is not visible.

Even if **current.commandLine** returns **off**, it is still possible for the command line to become visible. If you issue the **commandLine** action, the command line will become visible until focus is returned to edit area of the window.

Examples

```
query commandLine
set commandLine off
query current.commandLine
query default.commandLine
set default.commandLine off
query install.commandLine
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

current parameter

“default parameter” on page 83 parameter

“install parameter” on page 129 parameter
commandLine action

commands parameter

The **commands** parameter can be used to list all of the commands that are defined for the current view. Note that this list does not include the default commands unless they have been redefined.

Availability

query command

Scope

The current view.

Syntax

```
query commands
```

Examples

```
query commands
```

RELATED REFERENCES

“Default editor commands” on page 25

“query command” on page 47 command

“commandClass parameter” on page 71
parameter

compare.ignoreCase parameter

The **compare.ignoreCase** parameter determines whether differences in character case are significant when comparing files with the **compare** command.

Availability

query command

set command

Scope

compare.ignoreCase is scoped to the current view.

current.compare.ignoreCase is scoped to the current view.

default.compare.ignoreCase is globally scoped.

install.compare.ignoreCase is globally scoped.

Syntax

```
query compare.ignoreCase
set compare.ignoreCase [default | on | off]
query current.compare.ignoreCase
query default.compare.ignoreCase
set default.compare.ignoreCase [install | on | off]
query install.compare.ignoreCase
```


Parameters

default

If you specify the **default** parameter for the **set compare.ignoreCase** command, then the current view will use the value of **default.compare.ignoreCase** to determine if the **compare** command should be case sensitive.

on

If you specify the **on** parameter for the **set compare.ignoreCase** command, then the **compare** command will not be case sensitive for the current view. If you specify the **on** parameter for the **set default.compare.ignoreCase** command, then the **compare** command will not be case sensitive for all of the document views that have **compare.ignoreCase** set to **default**.

off

If you specify the **off** parameter for the **set compare.ignoreCase** command, then the **compare** command will be case sensitive for the current view. If you specify the **off** parameter for the **set default.compare.ignoreCase** command, then the **compare** command will be case sensitive for all of the document views that have **compare.ignoreCase** set to **default**.

install

If you specify the **install** parameter for the **set default.compare.ignoreCase** command, then all of the document views that have **compare.ignoreCase** set to **default** will use the value of **install.compare.ignoreCase** to determine if the **compare** command should not be case sensitive.

Description

The **query compare.ignoreCase** command will return the case sensitivity setting that will be used by the **compare** command.

If you do not specify any of the parameters for the **set compare.ignoreCase** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.compare.ignoreCase** command, then **install** is assumed.

The **query current.compare.ignoreCase** command will return **on** if, for the current view, the **compare** command will not be case sensitive and **off** if, for the current view, the **compare** command will be case sensitive.

Examples

```
query compare.ignoreCase
set compare.ignoreCase on
query current.compare.ignoreCase
query default.compare.ignoreCase
set default.compare.ignoreCase off
query install.compare.ignoreCase
```

RELATED REFERENCES

“compare command” on page 30 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“compare.ignoreLeadingBlanks parameter”
parameter
“compare.ignoreTrailingBlanks parameter” on
page 78 parameter
“compare.ignoreAllBlanks parameter” on
page 80 parameter

compare.ignoreLeadingBlanks parameter

The **compare.ignoreLeadingBlanks** parameter may be used to query or set whether differences in the number of blanks at the start of a line are significant when comparing files with the **compare** command.

Availability

query command
set command

Scope

compare.ignoreLeadingBlanks is scoped to the current view.
current.compare.ignoreLeadingBlanks is scoped to the current view.
default.compare.ignoreLeadingBlanks is globally scoped.
install.compare.ignoreLeadingBlanks is globally scoped.

Syntax

```
query compare.ignoreLeadingBlanks
set compare.ignoreLeadingBlanks [default | on | off]
query current.compare.ignoreLeadingBlanks
query default.compare.ignoreLeadingBlanks
set default.compare.ignoreLeadingBlanks [install | on | off]
query install.compare.ignoreLeadingBlanks
```

Parameters

default

If you specify the **default** parameter for the **set compare.ignoreLeadingBlanks** command, then the current view will use the value of **default.compare.ignoreLeadingBlanks** to determine if the **compare** command should ignore blanks at the beginning of the line.

on	If you specify the on parameter for the set compare.ignoreLeadingBlanks command, then the compare command will ignore the presence of blanks at the beginning of the line for the comparison in the current view. If you specify the on parameter for the set default.compare.ignoreLeadingBlanks command, then the compare command will ignore the presence of blanks at the beginning of the line for comparisons in any of the document views that have compare.ignoreLeadingBlanks set to default .
off	If you specify the off parameter for the set compare.ignoreLeadingBlanks command, then the compare command will not ignore the presence of blanks at the beginning of the line for the comparison in the current view. If you specify the off parameter for the set default.compare.ignoreLeadingBlanks command, then the compare command will not ignore the presence of blanks at the beginning of the line for comparisons in any of the document views that have compare.ignoreLeadingBlanks set to default .
install	If you specify the install parameter for the set default.compare.ignoreLeadingBlanks command, then all of the document views that have compare.ignoreLeadingBlanks set to default will use the value of install.compare.ignoreLeadingBlanks to determine if the compare command should ignore the presence of blanks at the beginning of the line.

Description

The **query compare.ignoreLeadingBlanks** command will return the leading blank comparison setting that will be used by the **compare** command.

If you do not specify any of the parameters for the **set compare.ignoreLeadingBlanks** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.compare.ignoreLeadingBlanks** command, then **install** is assumed.

The **query current.compare.ignoreLeadingBlanks** command will return **on** if, for the current view, the **compare** command will ignore blanks at the beginning of the line and **off** if, for the current view, the **compare** command will not ignore blanks at the beginning of the line.

Examples

```
query compare.ignoreLeadingBlanks
set compare.ignoreLeadingBlanks on
query current.compare.ignoreLeadingBlanks
query default.compare.ignoreLeadingBlanks
set default.compare.ignoreLeadingBlanks off
query install.compare.ignoreLeadingBlanks
```

RELATED REFERENCES

“compare command” on page 30 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“compare.ignoreCase parameter” on page 74 parameter
“compare.ignoreTrailingBlanks parameter” parameter
“compare.ignoreAllBlanks parameter” on page 80 parameter

compare.ignoreTrailingBlanks parameter

The **compare.ignoreTrailingBlanks** parameter may be used to query or set whether differences in the number of blanks at the end of a line are significant when comparing files with the **compare** command.

Availability

query command
set command

Scope

compare.ignoreTrailingBlanks is scoped to the current view.
current.compare.ignoreTrailingBlanks is scoped to the current view.
default.compare.ignoreTrailingBlanks is globally scoped.
install.compare.ignoreTrailingBlanks is globally scoped.

Syntax

```
query compare.ignoreTrailingBlanks
set compare.ignoreTrailingBlanks [default | on | off]
query current.compare.ignoreTrailingBlanks
query default.compare.ignoreTrailingBlanks
set default.compare.ignoreTrailingBlanks [install | on | off]
query install.compare.ignoreTrailingBlanks
```

Parameters

default

If you specify the **default** parameter for the **set compare.ignoreTrailingBlanks** command, then the current view will use the value of **default.compare.ignoreTrailingBlanks** to determine if the **compare** command should ignore blanks at the end of the line.

on	If you specify the on parameter for the set compare.ignoreTrailingBlanks command, then the compare command will ignore the presence of blanks at the end of the line for the comparison in the current view. If you specify the on parameter for the set default.compare.ignoreTrailingBlanks command, then the compare command will ignore the presence of blanks at the end of the line for comparisons in any of the document views that have compare.ignoreTrailingBlanks set to default .
off	If you specify the off parameter for the set compare.ignoreTrailingBlanks command, then the compare command will not ignore the presence of blanks at the end of the line for the comparison in the current view. If you specify the off parameter for the set default.compare.ignoreTrailingBlanks command, then the compare command will not ignore the presence of blanks at the end of the line for comparisons in any of the document views that have compare.ignoreTrailingBlanks set to default .
install	If you specify the install parameter for the set default.compare.ignoreTrailingBlanks command, then all of the document views that have compare.ignoreTrailingBlanks set to default will use the value of install.compare.ignoreTrailingBlanks to determine if the compare command should ignore the presence of blanks at the end of the line.

Description

The **query compare.ignoreTrailingBlanks** command will return the trailing blank comparison setting that will be used by the **compare** command.

If you do not specify any of the parameters for the **set compare.ignoreTrailingBlanks** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.compare.ignoreTrailingBlanks** command, then **install** is assumed.

The **query current.compare.ignoreTrailingBlanks** command will return **on** if, for the current view, the **compare** command will ignore blanks at the end of the line and **off** if, for the current view, the **compare** command will not ignore blanks at the end of the line.

Examples

```
query compare.ignoreTrailingBlanks
set compare.ignoreTrailingBlanks on
query current.compare.ignoreTrailingBlanks
query default.compare.ignoreTrailingBlanks
set default.compare.ignoreTrailingBlanks off
query install.compare.ignoreTrailingBlanks
```

RELATED REFERENCES

“compare command” on page 30 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“compare.ignoreCase parameter” on page 74 parameter
“compare.ignoreLeadingBlanks parameter” on page 76 parameter
“compare.ignoreAllBlanks parameter” parameter

compare.ignoreAllBlanks parameter

The **compare.ignoreAllBlanks** parameter may be used to query or set whether differences in the number of blanks in a line are significant when comparing files with the **compare** command.

Availability

query command
set command

Scope

compare.ignoreAllBlanks is scoped to the current view.
current.compare.ignoreAllBlanks is scoped to the current view.
default.compare.ignoreAllBlanks is globally scoped.
install.compare.ignoreAllBlanks is globally scoped.

Syntax

```
query compare.ignoreAllBlanks
set compare.ignoreAllBlanks [default | on | off]
query current.compare.ignoreAllBlanks
query default.compare.ignoreAllBlanks
set default.compare.ignoreAllBlanks [install | on | off]
query install.compare.ignoreAllBlanks
```

Parameters

default

If you specify the **default** parameter for the **set compare.ignoreAllBlanks** command, then the current view will use the value of **default.compare.ignoreAllBlanks** to determine if the **compare** command should ignore blanks in the line.

on

If you specify the **on** parameter for the **set compare.ignoreAllBlanks** command, then the **compare** command will ignore the presence of blanks in the line for the comparison in the current view. If you specify the **on** parameter for the **set default.compare.ignoreAllBlanks** command, then the **compare** command will ignore the presence of blanks in the line for comparisons in any of the document views that have **compare.ignoreAllBlanks** set to **default**.

off

If you specify the **off** parameter for the **set compare.ignoreAllBlanks** command, then the **compare** command will not ignore the presence of blanks in the line for the comparison in the current view. If you specify the **off** parameter for the **set default.compare.ignoreAllBlanks** command, then the **compare** command will not ignore the presence of blanks in the line for comparisons in any of the document views that have **compare.ignoreAllBlanks** set to **default**.

install

If you specify the **install** parameter for the **set default.compare.ignoreAllBlanks** command, then all of the document views that have **compare.ignoreAllBlanks** set to **default** will use the value of **install.compare.ignoreAllBlanks** to determine if the **compare** command should ignore the presence of blanks in the line.

Description

The **query compare.ignoreAllBlanks** command will return the blank comparison setting that will be used by the **compare** command.

If you do not specify any of the parameters for the **set compare.ignoreAllBlanks** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.compare.ignoreAllBlanks** command, then **install** is assumed.

The **query current.compare.ignoreAllBlanks** command will return **on** if, for the current view, the **compare** command will ignore blanks in the line and **off** if, for the current view, the **compare** command will not ignore blanks in the line.

Examples

```
query compare.ignoreAllBlanks
set compare.ignoreAllBlanks on
query current.compare.ignoreAllBlanks
query default.compare.ignoreAllBlanks
set default.compare.ignoreAllBlanks off
query install.compare.ignoreAllBlanks
```

RELATED REFERENCES

“compare command” on page 30 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“compare.ignoreCase parameter” on page 74 parameter
“compare.ignoreLeadingBlanks parameter”
on page 76 parameter
“compare.ignoreTrailingBlanks parameter” on
page 78 parameter

cursorRow parameter

The **cursorRow** parameter may be used to query or set the physical row of the line that contains the cursor.

Availability

query command
set command

Scope

The current view.

Syntax

```
query cursorRow  
set cursorRow n
```

Parameters

n

Use the *n* parameter to indicate the physical row number.

Description

Note that setting the physical row number will not change the current element. It will attempt to move the current element to the specified physical row number. It may not always be possible to completely comply with the specified physical row number. The current element must always be visible and elements are laid out such that there is never any wasted space at the top of the edit area and there is only wasted space at the bottom of the edit area if there are not enough visible elements to fill the edit area.

Examples

```
query cursorRow  
set cursorRow 3
```


RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“rowHeight parameter” on page 175 parameter
“rows parameter” on page 176 parameter

default parameter

The **default** parameter can be used to set the default setting of a parameter.

Some editor parameters may be set to default. This indicates that the value used for these parameters should be taken from the default setting of that parameter. For example, the **block.defaultType** parameter may be set to **default**, meaning that the default block type for the current view should be taken from the **default.block.defaultType** parameter. Changes made to the default parameter are saved in the editor profile.

Availability

query command
set command

Scope

The current view.

Syntax

```
query default.parameter  
set default.parameter setting
```

Parameters

parameter

Use the *parameter* qualifier to specify the name of the editor parameter for which you wish to determine its default setting. Note that not all editor parameters have a default setting.

setting

Use the *setting* parameter to indicate the desired setting. The syntax of the *setting* is dependent on *parameter*.

Examples

```
query default.block.defaultType  
set default.block.defaultType stream
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“block.defaultType parameter” on page 67 parameter
“commandLine parameter” on page 72 parameter
“compare.ignoreCase parameter” on page 74 parameter
“compare.ignoreLeadingBlanks parameter” on page 76 parameter
“compare.ignoreTrailingBlanks parameter” on page 78 parameter
“compare.ignoreAllBlanks parameter” on page 80 parameter
current parameter
“expandHide parameter” on page 90 parameter
“expandTabs parameter” on page 92 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“font parameter” on page 118 parameter parameter
“hideSequenceNumbers parameter” on page 125 parameter
“install parameter” on page 129 parameter
“lineNumbers parameter” on page 138 parameter
“messageLine parameter” on page 150 parameter
“popup parameter” on page 156 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
“save.textLimit parameter” on page 176 parameter
“save.trim parameter” on page 178 parameter
“sequenceNumbers parameter” on page 182 parameter
“statusLine parameter” on page 185 parameter
“tabs parameter” on page 189 parameter
“updateProfile.baseProfile parameter” on page 194 parameter
“updateProfile.extensions parameter” on page 196 parameter
“updateProfile.noParser parameter” on page 197 parameter
“updateProfile.palette parameter” on page 199 parameter
“updateProfile.paletteAttributes parameter” on page 201 parameter
“updateProfile.palettes parameter” on page 203 parameter
“updateProfile.parser parameter” on page 204 parameter
“updateProfile.parserAssociation parameter” on page 206 parameter
“updateProfile.parserClass parameter” on page 208 parameter
“updateProfile.parsers parameter” on page 210 parameter
“updateProfile.userActions parameter” on page 211 parameter
“updateProfile.userCommands parameter” on page 213 parameter
“updateProfile.userKeyActions parameter” on page 215 parameter
“updateProfile.userMouseActions parameter” on page 217 parameter
“updateProfile.userProfile parameter” on page 219 parameter

dirty parameter

The **dirty** parameter can be used to determine if there is an uncompleted change. The **dirty** parameter will return **on** if there is an uncompleted change or **off** if there is no uncompleted change.

Availability

query command

Scope

The current view.

Syntax

```
query dirty
```

Examples

```
query dirty
```

RELATED REFERENCES

“query command” on page 47 command

“save command” on page 50 command

“undo command” on page 54 command

“changes parameter” on page 69 parameter

displayPosition parameter

The **displayPosition** parameter may be used to query or set the cursor’s display column position on the current element. The display column position includes any spaces that have been added due to tab expansion and any characters that have been used to display the element’s sequence number. If there are no expanded tab characters and no sequence numbers then the **displayPosition** parameter behaves the same as the **position** parameter.

Availability

query command

set command

Scope

The current view.

Syntax

```
query displayPosition  
set displayPosition n
```

Parameters

n

Use the *n* parameter to indicate the display column position of the cursor.

Description

Note that it may not be possible to move the cursor to the specified display column position. If the specified display column position is in the middle of an expanded tab character or part of the sequence number, then the cursor will be moved to a display position that is not in the middle of an expanded tab character or part of a sequence number.

Examples

```
query displayPosition
set displayPosition 20
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“position parameter” on page 158 parameter

documentId parameter

The **documentId** parameter will return a unique integer that can be used to identify the document. Document identification numbers are not reused. If two document views return the same value for **documentId**, then they are different views on the same document.

Availability

query command

Scope

The current view.

Syntax

```
query documentId
```

Examples

```
query documentId
```

RELATED REFERENCES

“query command” on page 47 command

element parameter

The **element** parameter may be used to determine the ordinal number of the current element. An editor element is a line of editor text. Elements include show lines and non show lines. Show lines are not saved when the document is saved.

Availability

query command

Scope

The current view.

Syntax

```
query element
```

Examples

```
query element
```

RELATED REFERENCES

“locate command” on page 42 command
“query command” on page 47 command
“elements parameter” on page 88 parameter
“line parameter” on page 137 parameter
“lines parameter” on page 139 parameter
“show parameter” on page 184 parameter

elementClasses parameter

The **elementClasses** parameter may be used to query or set the classes for the current element. Classes must be registered with the **class** parameter before they can be specified for an element. Most parser assign classes to document elements.

Availability

query command
set command

Scope

The current element view.

Syntax

```
query elementClasses  
set elementClasses [ className ] [...]
```

Parameters

className

Use the *className* parameter to indicate the name of a class that you want to assign to the current element.

Description

You may specify any of the class names returned by the **classes** parameter. If you do not specify any classes, then all of the classes that are currently assigned to the element will be removed.

Examples

```
query elementClasses  
set elementClasses comment code
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“class parameter” on page 70 parameter
“classes parameter” on page 71 parameter
“excludedClasses parameter” on page 89 parameter
“includedClasses parameter” on page 127 parameter

elements parameter

The **elements** parameter may be used to determine the total number of elements in the current document. An editor element is a line of editor text. Elements include show lines and non show lines. Show lines are not saved when the document is saved.

Availability

query command

Scope

The current view.

Syntax

query elements

Examples

query elements

RELATED REFERENCES

“locate command” on page 42 command
“query command” on page 47 command
“element parameter” on page 86 parameter
“line parameter” on page 137 parameter
“lines parameter” on page 139 parameter
“show parameter” on page 184 parameter

emphasisLength parameter

The **emphasisLength** parameter may be used to query or set the number of characters that are emphasized. The emphasis is drawn at the current cursor location and removed when the cursor is moved. Some commands like **findText** and **locate** may set **emphasisLength** in order to highlight the results of the command.

Availability

query command
set command

Scope

The current view.

Syntax

```
query emphasisLength  
set emphasisLength n
```

Parameters

n

Use the *n* parameter to indicate the number of characters that should be emphasized.

Examples

```
query emphasisLength  
set emphasisLength 10
```

RELATED REFERENCES

“findText command” on page 34 command
“locate command” on page 42 command
“query command” on page 47 command
“set command” on page 52 command
“findText.emphasis parameter” on page 102 parameter

excludedClasses parameter

The **excludedClasses** parameter may be used to query or set the classes that should be excluded from the current view. Any elements which have been assigned an excluded class will not be visible in the current view.

Availability

query command
set command

Scope

The current view.

Syntax

```
query excludedClasses  
set excludedClasses [ className ] [...]
```

Parameters

className

Use the *className* parameter to indicate the name of a class that you want to exclude from the current element.

Description

You may specify any of the class names returned by the **classes** parameter. If you do not specify any classes, then none of the classes will be excluded from the current view.

Examples

```
query excludedClasses
set excludedClasses comment space
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“class parameter” on page 70 parameter
“classes parameter” on page 71 parameter
“elementClasses parameter” on page 87
parameter
“includedClasses parameter” on page 127
parameter
showAll action

expandHide parameter

The **expandHide** parameter may be used to set the visibility of the expand/hide area.

If **expandHide** is set to **on** and there are hidden elements in the current view, then the expand/hide area will be displayed at the left side of the screen. Visible elements that are followed by hidden elements will have a '+' character in the expand/hide area. If you click on the '+' character, the expanded parameter will be set to on and all of the hidden elements between the current element and the next hidden element will be displayed.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

expandHide is scoped to the current view.
current.expandHide is scoped to the current view.
default.expandHide is globally scoped.
install.expandHide is globally scoped.

Syntax

```
query expandHide
set expandHide [ default | on | off ]
query current.expandHide
query default.expandHide
set default.expandHide [ install | on | off ]
query install.expandHide
```


Parameters

default

If you specify the **default** parameter for the **set expandHide** command, then the current view will use the value of **default.expandHide** to determine the visibility of the expand/hide area.

on

If you specify the **on** parameter for the **set expandHide** command, then the expand/hide area will be visible on the current view. If you specify the **on** parameter for the **set default.expandHide** command, then the expand/hide area will be visible for all views that have **expandHide** set to **default**.

off

If you specify the **off** parameter for the **set expandHide** command, then the expand/hide area will not be visible on the current view. If you specify the **off** parameter for the **set default.expandHide** command, then the expand/hide area will not be visible for all views that have **expandHide** set to **default**.

install

If you specify the **install** parameter for the **set default.expandHide** command, then all of the views that have **expandHide** set to **default** will use the value of **install.expandHide** to determine the visibility of the expand/hide area.

Description

If you do not specify any of the parameters for the **set expandHide** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.expandHide** command, then **install** is assumed.

The **query current.expandHide** command will return **on** if the expand/hide area is visible or **off** if the expand/hide area is not visible.

Even if **current.expandHide** returns **on**, the expand/hide area will not be visible unless there are hidden elements in the current document view.

Examples

```
query expandHide
set expandHide off
query current.expandHide
query default.expandHide
set default.expandHide off
query install.expandHide
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
current parameter
“expandHideAreaWidth parameter”
parameter
“expanded parameter” on page 94 parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“topExpanded parameter” on page 193
parameter
expandHideAtMouse action

expandHideAreaWidth parameter

The **expandHideAreaWidth** parameter can be used to determine the width in pixels of the expand/hide area.

Availability

query command

Scope

The current view.

Syntax

```
query expandHideAreaWidth
```

Examples

```
query expandHideAreaWidth
```

RELATED REFERENCES

“query command” on page 47 command
“expandHide parameter” on page 90
parameter
“expanded parameter” on page 94 parameter
“prefixAreaWidth parameter” on page 158
parameter
“textAreaWidth parameter” on page 191
parameter

expandTabs parameter

The **expandTabs** parameter may be used to query or set if tab characters should be expanded or not.

Availability

query command

set command

current parameter
default parameter
install parameter

Scope

expandTabs is scoped to the current view.
current.expandTabs is scoped to the current view.
default.expandTabs is globally scoped.
install.expandTabs is globally scoped.

Syntax

```
query expandTabs
set expandTabs [ default | on | off ]
query current.expandTabs
query default.expandTabs
set default.expandTabs [ install | on | off ]
query install.expandTabs
```

Parameters

default

If you specify the **default** parameter for the **set expandTabs** command, then the current view will use the value of **default.expandTabs** to determine if tab characters should be expanded or not.

on

If you specify the **on** parameter for the **set expandTabs** command, then tab characters will be expanded in the current view. If you specify the **on** parameter for the **set default.expandTabs** command, then tab characters will be expanded in all views that have **expandTabs** set to **default**.

off

If you specify the **off** parameter for the **set expandTabs** command, then tab characters will not be expanded in the current view. If you specify the **off** parameter for the **set default.expandTabs** command, then tab characters will be expanded in all views that have **expandTabs** set to **default**.

install

If you specify the **install** parameter for the **set default.expandTabs** command, then all of the views that have **expandTabs** set to **default** will use the value of **install.expandTabs** to determine if tab characters should be expanded.

Description

If you do not specify any of the parameters for the **set expandTabs** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.expandTabs** command, then **install** is assumed.

The **query current.expandTabs** command will return **on** if tab characters are expanded in the current view or **off** if the tab characters are not expanded..

Examples

```
query expandTabs
set expandTabs off
query current.expandTabs
query default.expandTabs
set default.expandTabs off
query install.expandTabs
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
expandTabs action

expanded parameter

The **expanded** parameter may be used to query or set the visibility of hidden elements between the current element and the next visible element. If there are hidden elements in the document view, then you can set **expanded** for each of the visible elements that are followed by hidden elements. If you set **expanded to on**, then all of the hidden elements between the specified element and the next visible element will be forced visible. If you set **expanded** to off, then all of the hidden elements between the specified element and the next visible element will remain hidden.

Availability

query command
set command

Scope

The current element view.

Syntax

```
query expanded
set expanded { on | off }
```

Parameters

on

Use the **on** parameter to indicate that the hidden elements between the current element and the next visible element should be forced visible..

off

Use the **off** parameter to indicate that the hidden elements between the current element and the next visible element should remain hidden.

Description

The **expanded** attribute of all of the elements in the document view is set to **off** if the any of the following happens:

- the **includedClasses** parameter is altered.

- the **excludedClasses** parameter is altered.
- the **markIncluded** parameter is altered.
- the **markExcluded** parameter is altered.

Examples

```
query expanded
set expanded on
```

RELATED REFERENCES

“expandAll command” on page 33
 “query command” on page 47
 “set command” on page 52
 “excludedClasses parameter” on page 89
 “includedClasses parameter” on page 127
 “markIncluded parameter” on page 147
 “markExcluded parameter” on page 143
 “topExpanded parameter” on page 193

fields parameter

The **fields** parameter may be used to query or set the current editing fields. A field consists of one or more columns. When you insert or delete text in one field it does not affect the text in the fields that follow it.

Availability

query command
 set command

Scope

The current view.

Syntax

```
query fields
set fields [ startingColumn ] [...]
```

Parameters

startingColumn

Use the *startingColumn* parameter to indicate the starting column for a field. You may specify as many starting columns as you wish. Each starting column will indicate the start of a new field.

Examples

```
query fields
set fields 7 73
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

findText.asis parameter

The **findText.asis** parameter may be used to query or set the case sensitivity setting for the **findText** command.

Availability

query command

set command

current parameter

default parameter

install parameter

Scope

findText.asis is scoped to the current view.

current.findText.asis is scoped to the current view.

default.findText.asis is globally scoped.

install.findText.asis is globally scoped.

Syntax

```
query findText.asis
set findText.asis [ default | on | off ]
query current.findText.asis
query default.findText.asis
set default.findText.asis [ install | on | off ]
query install.findText.asis
```

Parameters

default

If you specify the **default** parameter for the **set findText.asis** command, then the current view will use the value of **default.findText.asis** for the case sensitivity setting for the **findText** command.

on

If you specify the **on** parameter for the **set findText.asis** command, then the **findText** command will search case sensitively within the current view. If you specify the **on** parameter for the set **default.findText.asis** command, then the **findText** command will search case sensitively within all of the document views that have **findText.asis** set to **default**.

off

If you specify the **off** parameter for the **set findText.asis** command, then the **findText** command will search case insensitively within the current view. If you specify the **off** parameter for the set **default.findText.asis** command, then the **findText** command will search case insensitively within all of the document views that have **findText.asis** set to **default**.

install

If you specify the **install** parameter for the **set default.findText.asis** command, then all of the views that have **findText.asis** set to **default** will use the value of **install.findText.asis** to determine if the **findText** command should search with or without case sensitivity.

Description

If you do not specify any of the parameters for the **set findText.asis** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.asis** command, then **install** is assumed.

The **query current.findText.asis** command will return **on** if, for the current view, the **findText** command is searching case sensitively and **off** if, for the current view, the **findText** command is searching case insensitively.

Examples

```
query findText.asis
set findText.asis off
query current.findText.asis
query default.findText.asis
set default.findText.asis on
query install.findText.asis
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“install parameter” on page 129 parameter

findText.block parameter

The **findText.block** parameter may be used to query or set if the **findText** command restricts its search to the selected text area.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.block is scoped to the current view.
current.findText.block is scoped to the current view.
default.findText.block is globally scoped.
install.findText.block is globally scoped.

Syntax

```
query findText.block
set findText.block [ default | on | off ]
query current.findText.block
query default.findText.block
set default.findText.block [ install | on | off ]
query install.findText.block
```

Parameters

default

If you specify the **default** parameter for the **set findText.block** command, then the current view will use the value of **default.findText.block** to decide if the **findText** command should restrict its search to the selected text.

on

If you specify the **on** parameter for the **set findText.block** command, then within the current view, the **findText** command will restrict its search to only the selected text. If you specify the **on** parameter for the **set default.findText.block** command, then within all of the document views that have **findText.block** set to **default**, the **findText** command will restrict its search to the selected text.

off

If you specify the **off** parameter for the **set findText.block** command, then within the current view, the **findText** command will not restrict its search to the selected text. If you specify the **off** parameter for the **set default.findText.block** command, then within all of the document views that have **findText.block** set to **default**, the **findText** command will not restrict its search to the selected text.

install

If you specify the **install** parameter for the **set default.findText.block** command, then all of the views that have **findText.block** set to **default** will use the value of **install.findText.block** to determine if the **findText** command should restrict its search to the selected text.

Description

If you do not specify any of the parameters for the **set findText.block** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.block** command, then **install** is assumed.

The **query current.findText.block** command will return **on** if, for the current view, the **findText** command is restricting its search to the selected text. The **query current.findText.block** command will return **off** if, for the current view, the **findText** command is not restricting its search to the selected text.

Examples

```
query findText.block
set findText.block off
query current.findText.block
query default.findText.block
set default.findText.block on
query install.findText.block
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“install parameter” on page 129 parameter

findText.columns parameter

The **findText.columns** parameter may be used to query or set if the **findText** command restricts its search to selected columns indicated by **current.findText.startColumn** and **current.findText.endColumn**.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.columns is scoped to the current view.
current.findText.columns is scoped to the current view.
default.findText.columns is globally scoped.
install.findText.columns is globally scoped.

Syntax

```
query findText.columns
set findText.columns [ default | on | off ]
query current.findText.columns
query default.findText.columns
set default.findText.columns [ install | on | off ]
query install.findText.columns
```

Parameters

default

If you specify the **default** parameter for the **set findText.columns** command, then the current view will use the value of **default.findText.columns** for the case sensitivity setting for the **findText** command. If you specify the **on** parameter for the **set findText.columns** command, then within the current view, the **findText** command will restrict its search to the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**. If you specify the **on** parameter for the **set default.findText.columns** command, then within all of the document views that have **findText.columns** set to **default**, the **findText** command will restrict its search to the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**.

on

off

If you specify the **off** parameter for the **set findText.columns** command, then within the current view, the **findText** command will not restrict its search to the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**. If you specify the **off** parameter for the **set default.findText.columns** command, then within all of the document views that have **findText.columns** set to **default**, the **findText** command will not restrict its search to the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**.

install

If you specify the **install** parameter for the **set default.findText.columns** command, then all of the views that have **findText.columns** set to **default** will use the value of **install.findText.columns** to determine if the **findText** command should restrict its search to the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**.

Description

If you do not specify any of the parameters for the **set findText.columns** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.columns** command, then **install** is assumed.

The **query current.findText.columns** command will return **on** if the **findText** command is restricting its search to the text found between the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**. The **query current.findText.columns** command will return **off** if the **findText** command is not restricting its search to the text found between the columns indicated by **current.findText.startColumn** and **current.findText.endColumn**.

Examples

```
query findText.columns
set findText.columns off
query current.findText.columns
query default.findText.columns
set default.findText.columns on
query install.findText.columns
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96
parameter
“findText.block parameter” on page 98
parameter
“findText.emphasis parameter” parameter
“findText.endColumn parameter” on
page 104 parameter
“findText.findText parameter” on page 106
parameter
“findText.mark parameter” on page 108
parameter
“findText.regularExpression parameter” on
page 110 parameter
“findText.replaceText parameter” on page 112
parameter
“findText.startColumn parameter” on
page 114 parameter
“findText.wrap parameter” on page 116
parameter
“install parameter” on page 129 parameter

findText.emphasis parameter

The **findText.emphasis** parameter may be used to query or set if the **findText** command will emphasize the found text.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.emphasis is scoped to the current view.
current.findText.emphasis is scoped to the current view.
default.findText.emphasis is globally scoped.
install.findText.emphasis is globally scoped.

Syntax

```
query findText.emphasis
set findText.emphasis [ default | on | off ]
query current.findText.emphasis
query default.findText.emphasis
set default.findText.emphasis [ install | on | off ]
query install.findText.emphasis
```

Parameters

default

If you specify the **default** parameter for the **set findText.emphasis** command, then the current view will use the value of **default.findText.emphasis** to decide if the **findText** command should emphasize the found text.

on

If you specify the **on** parameter for the **set findText.emphasis** command, then within the current view, the **findText** command will emphasize the found text. If you specify the **on** parameter for the **set default.findText.emphasis** command, then within all the views that have **findText.emphasis** set to **default**, the **findText** command will emphasize the found text.

off

If you specify the **off** parameter for the **set findText.emphasis** command, then within the current view, the **findText** command will not emphasize the found text. If you specify the **off** parameter for the **set default.findText.emphasis** command, then within all the views that have **findText.emphasis** set to **default**, the **findText** command will not emphasize the found text.

install

If you specify the **install** parameter for the **set default.findText.emphasis** command, then all of the views that have **findText.emphasis** set to **default** will use the value of **install.findText.emphasis** to determine if the **findText** command should emphasize the found text.

Description

If you do not specify any of the parameters for the **set findText.emphasis** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.emphasis** command, then **install** is assumed.

The **query current.findText.emphasis** command will return **on** if, for the current view, the **findText** command is emphasizing the found text. The **query current.findText.emphasis** command will return **off** if, for the current view, the **findText** command is not emphasizing the found text.

Examples

```
query findText.emphasis
set findText.emphasis off
query current.findText.emphasis
query default.findText.emphasis
set default.findText.emphasis on
query install.findText.emphasis
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“emphasisLength parameter” on page 88 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.endColumn parameter” parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“install parameter” on page 129 parameter

findText.endColumn parameter

The **findText.endColumn** parameter may be used to query or set the end column that will be used by the **findText** command if the search is restricted to columns by **current.findText.columns**.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.endColumn is scoped to the current view.
current.findText.endColumn is scoped to the current view.
default.findText.endColumn is globally scoped.
install.findText.endColumn is globally scoped.

Syntax

```
query findText.endColumn
set findText.endColumn [ default | n ]
query current.findText.endColumn
query default.findText.endColumn
set default.findText.endColumn [ install | n ]
query install.findText.endColumn
```

Parameters

default

If you specify the **default** parameter for the **set findText.endColumn** command, then the current view will use the value of **default.findText.endColumn** for column restricted searches with the **findText** command.

n

If you specify the *n* parameter for the **set findText.endColumn** command, then within the current view, the **findText** command will use *n* as the end column for column restricted searches. If you specify the *n* parameter for the **set default.findText.endColumn** command, then within all document views that have the **findText.endColumn** set to **default**, the **findText** command will use *n* as the end column for column restricted searches.

install

If you specify the **install** parameter for the **set default.findText.endColumn** command, then all of the views that have **findText.endColumn** set to **default** will use the value of **install.findText.endColumn** to determine if the end column that should be used by the **findText** command for column restricted searches.

Description

If you do not specify any of the parameters for the **set findText.endColumn** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.endColumn** command, then **install** is assumed.

The **query current.findText.endColumn** command will return the end column that will be used, within the current view, by the **findText** command for column restricted searches.

Examples

```
query findText.endColumn
set findText.endColumn 80
query current.findText.endColumn
query default.findText.endColumn
set default.findText.endColumn 100
query install.findText.endColumn
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96
parameter
“findText.block parameter” on page 98
parameter
“findText.columns parameter” on page 100
parameter
“findText.emphasis parameter” on page 102
parameter
“findText.findText parameter” parameter
“findText.mark parameter” on page 108
parameter
“findText.regularExpression parameter” on
page 110 parameter
“findText.replaceText parameter” on page 112
parameter
“findText.startColumn parameter” on
page 114 parameter
“findText.wrap parameter” on page 116
parameter
“install parameter” on page 129 parameter

findText.findText parameter

The **findText.findText** parameter may be used to query or set the text to be found by the **findText** command.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.findText is scoped to the current view.
current.findText.findText is scoped to the current view.
default.findText.findText is globally scoped.
install.findText.findText is globally scoped.

Syntax

```
query findText.findText
set findText.findText [ default | "text" ]
query current.findText.findText
query default.findText.findText
set default.findText.findText [ install | "text" ]
query install.findText.findText
```


Parameters

default

If you specify the **default** parameter for the **set findText.findText** command, then the current view will use the value of **default.findText.findText** as the search text for the **findText** command.

text

If you specify the *text* parameter for the **set findText.findText** command, then within the current view, the **findText** command will search for *text*. If you specify the *text* parameter for the **set default.findText.findText** command, then within all views that have **findText.findText** set to **default**, the **findText** command will search for *text*. If you are quoting the *text* parameter then you will need to ensure that all backslashes and imbedded quotes are quoted with a backslash. e.g. to quote **a"b\c** you need to specify **"a\"b\\c"**.

install

If you specify the **install** parameter for the **set default.findText.findText** command, then within all of document views that have **findText.findText** set to **default**, the **findText** command will search for the value of **install.findText.findText**.

Description

If you do not specify any of the parameters for the **set findText.findText** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.findText** command, then **install** is assumed.

The **query current.findText.findText** command will return the text that the **findText** will search for within the current view.

Examples

```
query findText.findText
set findText.findText "hello"
query current.findText.findText
query default.findText.findText
set default.findText.findText "hello"
query install.findText.findText
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96
parameter
“findText.block parameter” on page 98
parameter
“findText.columns parameter” on page 100
parameter
“findText.emphasis parameter” on page 102
parameter
“findText.endColumn parameter” on
page 104 parameter
“findText.mark parameter” parameter
“findText.regularExpression parameter” on
page 110 parameter
“findText.replaceText parameter” on page 112
parameter
“findText.startColumn parameter” on
page 114 parameter
“findText.wrap parameter” on page 116
parameter
“install parameter” on page 129 parameter

findText.mark parameter

The **findText.mark** parameter may be used to query or set if the **findText** command will select the found text.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.mark is scoped to the current view.
current.findText.mark is scoped to the current view.
default.findText.mark is globally scoped.
install.findText.mark is globally scoped.

Syntax

```
query findText.mark
set findText.mark [ default | on | off ]
query current.findText.mark
query default.findText.mark
set default.findText.mark [ install | on | off ]
query install.findText.mark
```

Parameters

default

If you specify the **default** parameter for the **set findText.mark** command, then the current view will use the value of **default.findText.mark** to decide if the **findText** command should select the found text.

on

If you specify the **on** parameter for the **set findText.mark** command, then within the current view, the **findText** command will select the found text. If you specify the **on** parameter for the **set default.findText.mark** command, then within all the views that have **findText.mark** set to **default**, the **findText** command will select the found text.

off

If you specify the **off** parameter for the **set findText.mark** command, then within the current view, the **findText** command will not select the found text. If you specify the **off** parameter for the **set default.findText.mark** command, then within all the views that have **findText.mark** set to **default**, the **findText** command will not select the found text.

install

If you specify the **install** parameter for the **set default.findText.mark** command, then all of the views that have **findText.mark** set to **default** will use the value of **install.findText.mark** to determine if the **findText** command should select the found text.

Description

If you do not specify any of the parameters for the **set findText.mark** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.mark** command, then **install** is assumed.

The **query current.findText.mark** command will return **on** if, for the current view, the **findText** command is selecting the found text. The **query current.findText.mark** command will return **off** if, for the current view, the **findText** command is not selecting the found text.

Examples

```
query findText.mark
set findText.mark off
query current.findText.mark
query default.findText.mark
set default.findText.mark on
query install.findText.mark
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
parameter
“findText.asis parameter” on page 96
parameter
“findText.block parameter” on page 98
parameter
“findText.columns parameter” on page 100
parameter
“findText.emphasis parameter” on page 102
parameter
“findText.endColumn parameter” on
page 104 parameter
“findText.findText parameter” on page 106
parameter
“findText.regularExpression parameter”
parameter
“findText.replaceText parameter” on page 112
parameter
“findText.startColumn parameter” on
page 114 parameter
“findText.wrap parameter” on page 116
parameter
“install parameter” on page 129 parameter

findText.regularExpression parameter

The **findText.regularExpression** parameter may be used to query or set the regular expression setting for the **findText** command.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.regularExpression is scoped to the current view.
current.findText.regularExpression is scoped to the current view.
default.findText.regularExpression is globally scoped.
install.findText.regularExpression is globally scoped.

Syntax

```
query findText.regularExpression
set findText.regularExpression [ default | on | off ]
query current.findText.regularExpression
query default.findText.regularExpression
set default.findText.regularExpression [ install | on | off ]
query install.findText.regularExpression
```

Parameters

default

If you specify the **default** parameter for the **set findText.regularExpression** command, then the current view will use the value of **default.findText.regularExpression** for the regular expression setting for the **findText** command.

on

If you specify the **on** parameter for the **set findText.regularExpression** command, then the **findText** command will use regular expression pattern matching while searching within the current view. If you specify the **on** parameter for the **set default.findText.regularExpression** command, then the **findText** command will use regular expression pattern matching while searching within all of document views that have **findText.regularExpression** set to **default**.

off

If you specify the **off** parameter for the **set findText.regularExpression** command, then the **findText** command will not use regular expression pattern matching while searching within the current view. If you specify the **off** parameter for the **set default.findText.regularExpression** command, then the **findText** command will not use regular expression pattern matching while searching within all of document views that have **findText.regularExpression** set to **default**.

install

If you specify the **install** parameter for the **set default.findText.regularExpression** command, then all of the views that have **findText.regularExpression** set to **default** will use the value of **install.findText.regularExpression** to determine if the **findText** command should use regular expression pattern matching to perform its searches.

Description

If you do not specify any of the parameters for the **set findText.regularExpression** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.regularExpression** command, then **install** is assumed.

The **query current.findText.regularExpression** command will return **on** if the **findText** command is using regular expression pattern matching to perform searches within the current view.

Examples

```
query findText.regularExpression
set findText.regularExpression off
query current.findText.regularExpression
```

```
query default.findText.regularExpression
set default.findText.regularExpression on
query install.findText.regularExpression
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.replaceText parameter” parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“install parameter” on page 129 parameter

findText.replaceText parameter

The **findText.replaceText** parameter may be used to query or set the replacement text that will be used by the **findText** command.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.replaceText is scoped to the current view.
current.findText.replaceText is scoped to the current view.
default.findText.replaceText is globally scoped.
install.findText.replaceText is globally scoped.

Syntax

```
query findText.replaceText
set findText.replaceText [ default | "text" ]
query current.findText.replaceText
query default.findText.replaceText
set default.findText.replaceText [ install | "text" ]
query install.findText.replaceText
```

Parameters

default

If you specify the **default** parameter for the **set findText.replaceText** command, then the current view will use the value of **default.findText.replaceText** as the replacement text for the **findText** command.

text

If you specify the *text* parameter for the **set findText.replaceText** command, then within the current view, the **findText** command will use *text* as the replacement text. If you specify the *text* parameter for the **set default.findText.replaceText** command, then within all views that have **findText.replaceText** set to **default**, the **findText** command will use *text* as the replacement text. If you are quoting the *text* parameter then you will need to ensure that all backslashes and imbedded quotes are quoted with a backslash. e.g. to quote **a"b\c** you need to specify **a\"b\\c"**.

install

If you specify the **install** parameter for the **set default.findText.replaceText** command, then within all of document views that have **findText.replaceText** set to **default**, the **findText** command will use the the value of **install.findText.replaceText** as the replacement text.

Description

If you do not specify any of the parameters for the **set findText.replaceText** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.replaceText** command, then **install** is assumed.

The **query current.findText.replaceText** command will return the replacement text that the **findText** will use within the current view.

Examples

```
query findText.replaceText
set findText.replaceText "hello"
query current.findText.replaceText
query default.findText.replaceText
set default.findText.replaceText "hello"
query install.findText.replaceText
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” parameter
“findText.wrap parameter” on page 116 parameter
“install parameter” on page 129 parameter

findText.startColumn parameter

The **findText.startColumn** parameter may be used to query or set the start column that will be used by the **findText** command if the search is restricted to columns by **current.findText.columns**.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.startColumn is scoped to the current view.
current.findText.startColumn is scoped to the current view.
default.findText.startColumn is globally scoped.
install.findText.startColumn is globally scoped.

Syntax

```
query findText.startColumn
set findText.startColumn [ default | n ]
query current.findText.startColumn
query default.findText.startColumn
set default.findText.startColumn [ install | n ]
query install.findText.startColumn
```


Parameters

default

If you specify the **default** parameter for the **set findText.startColumn** command, then the current view will use the value of **default.findText.startColumn** for column restricted searches with the **findText** command.

n

If you specify the *n* parameter for the **set findText.startColumn** command, then within the current view, the **findText** command will use *n* as the start column for column restricted searches. If you specify the *n* parameter for the **set default.findText.startColumn** command, then within all document views that have the **findText.startColumn** set to **default**, the **findText** command will use *n* as the start column for column restricted searches.

install

If you specify the **install** parameter for the **set default.findText.startColumn** command, then all of the views that have **findText.startColumn** set to **default** will use the value of **install.findText.startColumn** to determine if the start column that should be used by the **findText** command for column restricted searches.

Description

If you do not specify any of the parameters for the **set findText.startColumn** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.startColumn** command, then **install** is assumed.

The **query current.findText.startColumn** command will return the start column that will be used, within the current view, by the **findText** command for column restricted searches.

Examples

```
query findText.startColumn
set findText.startColumn 1
query current.findText.startColumn
query default.findText.startColumn
set default.findText.startColumn 7
query install.findText.startColumn
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96
parameter
“findText.block parameter” on page 98
parameter
“findText.columns parameter” on page 100
parameter
“findText.emphasis parameter” on page 102
parameter
“findText.endColumn parameter” on
page 104 parameter
“findText.findText parameter” on page 106
parameter
“findText.mark parameter” on page 108
parameter
“findText.regularExpression parameter” on
page 110 parameter
“findText.replaceText parameter” on page 112
parameter
“findText.wrap parameter” parameter
“install parameter” on page 129 parameter

findText.wrap parameter

The **findText.wrap** parameter may be used to query or set the wrap setting for the **findText** command.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

findText.wrap is scoped to the current view.
current.findText.wrap is scoped to the current view.
default.findText.wrap is globally scoped.
install.findText.wrap is globally scoped.

Syntax

```
query findText.wrap
set findText.wrap [ default | on | off ]
query current.findText.wrap
query default.findText.wrap
set default.findText.wrap [ install | on | off ]
query install.findText.wrap
```

Parameters

default

If you specify the **default** parameter for the **set findText.wrap** command, then the current view will use the value of **default.findText.wrap** for the wrap setting for the **findText** command.

on

If you specify the **on** parameter for the **set findText.wrap** command, then the **findText** command will wrap around to the top (bottom) of the file when searching down (up) within the current view. If you specify the **on** parameter for the **set default.findText.wrap** command, then the **findText** command will wrap around to the top (bottom) of the file when searching down (up) within all of document views that have **findText.wrap** set to **default**.

off

If you specify the **off** parameter for the **set findText.wrap** command, then the **findText** command will not wrap around to the top (bottom) of the file when searching down (up) within the current view. If you specify the **off** parameter for the **set default.findText.wrap** command, then the **findText** command will not wrap around to the top (bottom) of the file when searching down (up) within all of document views that have **findText.wrap** set to **default**.

install

If you specify the **install** parameter for the **set default.findText.wrap** command, then all of the views that have **findText.wrap** set to **default** will use the value of **install.findText.wrap** to determine if the **findText** command should wrap around to the top (bottom) of the file when searching down (up).

Description

If you do not specify any of the parameters for the **set findText.wrap** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.findText.wrap** command, then **install** is assumed.

The **query current.findText.wrap** command will return **on** if the **findText** command will wrap around to the top (bottom) of the file when searching down (up) and **off** if the **findText** command will not wrap when it hits the top or bottom of the file.

Examples

```
query findText.wrap
set findText.wrap off
query current.findText.wrap
query default.findText.wrap
set default.findText.wrap on
query install.findText.wrap
```

RELATED REFERENCES

“findText command” on page 34 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“findText.asis parameter” on page 96
parameter
“findText.block parameter” on page 98
parameter
“findText.columns parameter” on page 100
parameter
“findText.emphasis parameter” on page 102
parameter
“findText.endColumn parameter” on
page 104 parameter
“findText.findText parameter” on page 106
parameter
“findText.mark parameter” on page 108
parameter
“findText.regularExpression parameter” on
page 110 parameter
“findText.replaceText parameter” on page 112
parameter
“findText.startColumn parameter” on
page 114 parameter
“install parameter” on page 129 parameter

font parameter

The **font** parameter may be used to query or set the current font used to display text in the edit window.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

font is scoped to the current view.
current.font is scoped to the current view.
default.font is globally scoped.
install.font is globally scoped.

Syntax

```
query font
set font [ default | "fontName" ]
query current.font
query default.font
set default.font [ install | "fontName" ]
query install.font
```

Parameters

`default`

If you specify the **default** parameter for the **set font** command, then the current view will use the value of **default.font** to determine the font that should be used to display text in the edit window.

fontName

If you specify the *fontName* parameter for the **set font** command, then the font indicated by *fontName* will be used by the current view. If you specify the *fontName* parameter for the **set default.font** command, then the font indicated by *fontName* will be used by all views that have **font** set to **default**. The *fontName* parameter should be in one of the following forms:

- *fontname-style-pointsize*
- *fontname-pointsize*
- *fontname-style*
- *fontname*

where *style* is one of **bold**, **bolditalic**, or **italic** and *pointsize* is a decimal representation of the point size.

`install`

If you specify the **install** parameter for the **set default.font** command, then all of the views that have **font** set to **default** will use the value of **install.font** to determine the font that should be used to display text in the edit window.

Description

If you do not specify any of the parameters for the **set font** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.font** command, then **install** is assumed.

The **query current.font** command will return the font that is being used by the current view.

Examples

```
query font
set font "Monospaced-12"
query current.font
query default.font
set default.font "Monospaced-bold-14"
query install.font
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

current parameter

“default parameter” on page 83 parameter

“install parameter” on page 129 parameter

forceAllVisible parameter

The **forceAllVisible** parameter may be used to query or set the forced visibility of all of the document elements. You may use this parameter to ensure that all document elements are always visible in the current view.

Availability

query command
set command

Scope

The current view.

Syntax

```
query forceAllVisible  
set forceAllVisible { on | off }
```

Parameters

on

Use the **on** parameter to indicate that the all document elements should always be visible.

off

Use the **off** parameter to indicate that you wish to allow the document elements to become invisible.

Examples

```
query forceAllVisible  
set forceAllVisible on
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“excludedClasses parameter” on page 89 parameter

“forceVisible parameter” parameter

“includedClasses parameter” on page 127 parameter

“markIncluded parameter” on page 147 parameter

“markExcluded parameter” on page 143 parameter

forceVisible parameter

The **forceVisible** parameter may be used to query or set the forced visibility of an element. You may use this parameter to ensure that an element is always visible.

Availability

query command
set command

Scope

The current view’s view of the current element.

Syntax

```
query forceVisible  
set forceVisible { on | off }
```

Parameters

on

Use the **on** parameter to indicate that the current element should always be visible.

off

Use the **off** parameter to indicate that you wish to allow the current element to become invisible.

Examples

```
query forceVisible  
set forceVisible on
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“excludedClasses parameter” on page 89 parameter

“forceAllVisible parameter” on page 120 parameter

“includedClasses parameter” on page 127 parameter

“markIncluded parameter” on page 147 parameter

“markExcluded parameter” on page 143 parameter

headerMark parameter

The **headerMark** parameter may be used to determine if the current element is a header element for an excluded mark.

If the **markExcludedHeader** parameter for a mark is **on**, then the specified mark will have a header element when it is excluded. The header element is a **show** element (will not be saved with the file) and contains text that indicates how many lines are excluded. The **headerMark** parameter will return the name of the mark for which the current element is a header mark. If the current element is not a header mark, then the **headerMark** parameter will return **null**.

Availability

query command

Scope

The current element of the current view.

Syntax

```
query headerMark
```

Examples

```
query headerMark
```

RELATED REFERENCES

“query command” on page 47 command
“mark parameter” on page 141 parameter
“markExcluded parameter” on page 143
parameter
“markExcludedHeader parameter” on
page 144 parameter

help.configuration parameter

The **help.configuration** parameter may be used to query or set the configuration file that will be used by the **help** command. The help configuration file governs the setup of the html help system. Without a correct help configuration file, it is not possible to view the help for lpex.

Availability

query command
set command

Scope

help.configuration is globally scoped.

Syntax

```
query help.configuration  
set help.configuration
```

Description

The **query help.configuration** command will return the help configuration file that will be used by the **help** command.

The configuration file specified must be specified using a full path name.

If a configuration file is specified that cannot be understood by the help system, the editor will behave as though no configuration file were specified, in which case no help will be shown.

Examples

```
query help.configuration  
set help.configuration f:\help.cfg
```

RELATED REFERENCES

“help command” on page 38 command
“help.location parameter” on page 123
parameter
“help.homepage parameter” on page 123
parameter
“query command” on page 47 command
“set command” on page 52 command

help.homepage parameter

The **help.homepage** parameter may be used to query or set the default help page that will be displayed when no other help is available for a help request. Without a correct default help page, no help will be shown when a help request is made and no help is available.

Availability

query command
set command

Scope

help.homepage is globally scoped.

Syntax

```
query help.homepage
set help.homepage
```

Description

The **query help.homepage** command will return the help page that will be shown when no other help is available.

The page specified must be available to the help system. It must be part of the help files registered during installation.

If a default help page is specified that cannot be found by the help system, the browser will indicate that the file could not be found.

Examples

```
query help.homepage
set help.homepage defaultPage.html
```

RELATED REFERENCES

“help command” on page 38 command
“help.configuration parameter” on page 122
parameter
“help.location parameter” parameter
“query command” on page 47 command
“set command” on page 52 command

help.location parameter

The **help.location** parameter may be used to query or set the location of properties files used to map help requests to the appropriate html pages. Without a correct help location, it is not possible to view the help for lpex.

Availability

query command
set command

Scope

help.location is globally scoped.

Syntax

```
query help.location  
set help.location
```

Description

The **query help.location** command will return the location of the html help mapping files

The location specified must be specified using a full path name.

If a location is specified that cannot be understood by the help system, the editor will behave as though no location were specified, in which case no help will be shown.

Examples

```
query help.location  
set help.location f:\help
```

RELATED REFERENCES

“help command” on page 38 command
“help.configuration parameter” on page 122
parameter
“help.homepage parameter” on page 123
parameter
“query command” on page 47 command
“set command” on page 52 command

hex parameter

The **hex** parameter may be used to query the hexadecimal value of the character at the current cursor location.

Availability

query command

Scope

The character at the current position of the current element of the current view.

Syntax

```
query hex
```

Examples

```
query hex
```

RELATED REFERENCES

“query command” on page 47 command
“text parameter” on page 191 parameter

hideSequenceNumbers parameter

The **hideSequenceNumbers** parameter may be used to query or set if the document's sequence numbers should be visible or not.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

hideSequenceNumbers is scoped to the current view.
current.hideSequenceNumbers is scoped to the current view.
default.hideSequenceNumbers is globally scoped.
install.hideSequenceNumbers is globally scoped.

Syntax

```
query hideSequenceNumbers
set hideSequenceNumbers [ default | on | off ]
query current.hideSequenceNumbers
query default.hideSequenceNumbers
set default.hideSequenceNumbers [ install | on | off ]
query install.hideSequenceNumbers
```

Parameters

default

If you specify the **default** parameter for the **set hideSequenceNumbers** command, then the current view will use the value of **default.hideSequenceNumbers** to determine if the document's sequence numbers should be visible or not.

on

If you specify the **on** parameter for the **set hideSequenceNumbers** command, then the document's sequence numbers will be visible in the current view. If you specify the **on** parameter for the **set default.hideSequenceNumbers** command, then the document's sequence numbers will be visible in all document views that have **hideSequenceNumbers** set to **default**.

off

If you specify the **off** parameter for the **set hideSequenceNumbers** command, then the document's sequence numbers will be hidden in the current view. If you specify the **off** parameter for the **set default.hideSequenceNumbers** command, then the document's sequence numbers will be hidden in all document views that have **hideSequenceNumbers** set to **default**.

install

If you specify the **install** parameter for the **set default.hideSequenceNumbers** command, then all of the views that have **hideSequenceNumbers** set to **default** will use the value of **install.hideSequenceNumbers** to determine if the document's sequence numbers should be visible or not.

Description

If you do not specify any of the parameters for the **set hideSequenceNumbers** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.hideSequenceNumbers** command, then **install** is assumed.

The **query current.hideSequenceNumbers** command will return **on** if the document's sequence numbers are visible and **off** if the document's sequence numbers are hidden.

Examples

```
query hideSequenceNumbers
set hideSequenceNumbers off
query current.hideSequenceNumbers
query default.hideSequenceNumbers
set default.hideSequenceNumbers off
query install.hideSequenceNumbers
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

current parameter

“default parameter” on page 83 parameter

“install parameter” on page 129l parameter

“sequenceNumber parameter” on page 180

parameter

“sequenceNumbers parameter” on page 182
parameter

inPrefix parameter

The **inPrefix** parameter may be used to query or set the cursor location in the prefix or text area.

Availability

query command

set command

Scope

The current view.

Syntax

```
query inPrefix
set inPrefix { on | off }
```

Parameters

on

If you specify the **on** parameter then the cursor will be moved into the prefix area.

off

If you specify the **off** parameter then the cursor will be moved into the text area.

Description

The prefix area is the an area to the left of the text area that is visible when the **current.lineNumbers** parameter is **on**. In some of the editor base behaviours this area can be used to enter prefix commands. Note that you will not be able to set **inPrefix** to **on** if the prefix area is not visible or if **prefixProtect** is set to **on**.

Examples

```
query inPrefix  
set inPrefix on
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“processPrefix command” on page 46 command

“baseProfile parameter” on page 66 parameter

“lineNumbers parameter” on page 138 parameter

“prefixAreaWidth parameter” on page 158 parameter

“prefixPosition parameter” on page 159 parameter

“prefixProtect parameter” on page 160 parameter

“prefixText parameter” on page 161 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

prefixBackSpace action

prefixDelete action

prefixEnd action

prefixHome action

prefixLeft action

prefixRight action

prefixTruncate action

processPrefix action

includedClasses parameter

The **includedClasses** parameter may be used to query or set the classes that should be included in the current view. If you specify one or more included classes, then any elements which have not been assigned one of the included classes will not be visible in the current view.

Availability

query command

set command

Scope

The current view.

Syntax

```
query includedClasses  
set includedClasses [ className ] [...]
```

Parameters

className

Use the *className* parameter to indicate the name of a class that you want to include from the current element.

Description

You may specify any of the class names returned by the **classes** parameter. If you do not specify any classes, then all of the classes will be included in the current view.

Examples

```
query includedClasses  
set includedClasses comment
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“class parameter” on page 70 parameter
“classes parameter” on page 71 parameter
“elementClasses parameter” on page 87 parameter
“excludedClasses parameter” on page 89 parameter
showAll action

insertMode parameter

The **insertMode** parameter may be used to query or set the current insertion mode.

Availability

query command
set command

Scope

The current view.

Syntax

```
query insertMode  
set insertMode { on | off }
```

Parameters

on

If you specify the **on** parameter then the insertion mode will be changed to insert.

off

If you specify the **off** parameter then the insertion mode will be changed to replace.

Description

If you are typing and the insertion mode is insert, then characters to the right of the cursor will be shifted to make room for the new characters. If you are typing and the insertion mode is replace, then characters will be replaced by the new characters.

Examples

```
query insertMode
set insertMode on
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
toggleInsert action

install parameter

The **install** parameter may be used to determine the installation setting of a parameter. If the **default** setting for a parameter is set to **install**, then the installation setting of the parameter will be used as the default setting. For example, if the **default.block.defaultType** parameter is set to **install**, the installation default block type should be used for all of the document views that have the **block.defaultType** set to **default**.

Availability

query command

Scope

Global.

Syntax

```
query install.parameter
```

Parameters

parameter

Use the *parameter* qualifier to specify the name of the editor parameter for which you wish to determine its install setting. Note that not all editor parameters have a install setting.

Description

Application writers may want to change the installation settings that are used by the editor. This can be accomplished by specifying an alternative installation profile with the **installProfile** parameter.

Examples

```
query install.block.defaultType
```

RELATED REFERENCES

“query command” on page 47 command
“block.defaultType parameter” on page 67 parameter
“commandLine parameter” on page 72 parameter
“compare.ignoreCase parameter” on page 74 parameter
“compare.ignoreLeadingBlanks parameter” on page 76 parameter
“compare.ignoreTrailingBlanks parameter” on page 78 parameter
“compare.ignoreAllBlanks parameter” on page 80 parameter
current parameter
“default parameter” on page 83 parameter
“expandHide parameter” on page 90 parameter
“expandTabs parameter” on page 92 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“font parameter” on page 118 parameter
“hideSequenceNumbers parameter” on page 125 parameter
“installProfile parameter” on page 131 parameter
“lineNumbers parameter” on page 138 parameter
“messageLine parameter” on page 150 parameter
“popup parameter” on page 156 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
“save.textLimit parameter” on page 176 parameter
“save.trim parameter” on page 178 parameter
“sequenceNumbers parameter” on page 182 parameter
“statusLine parameter” on page 185 parameter
“tabs parameter” on page 189 parameter
“updateProfile.baseProfile parameter” on page 194 parameter
“updateProfile.extensions parameter” on page 196 parameter
“updateProfile.noParser parameter” on page 197 parameter
“updateProfile.palette parameter” on page 199 parameter
“updateProfile.paletteAttributes parameter” on page 201 parameter
“updateProfile.palettes parameter” on page 203 parameter
“updateProfile.parser parameter” on page 204 parameter
“updateProfile.parserAssociation parameter” on page 206 parameter
“updateProfile.parserClass parameter” on page 208 parameter
“updateProfile.parsers parameter” on page 210 parameter
“updateProfile.userActions parameter” on page 211 parameter
“updateProfile.userCommands parameter” on page 213 parameter
“updateProfile.userKeyActions parameter” on page 215 parameter
“updateProfile.userMouseActions parameter” on page 217 parameter
“updateProfile.userProfile parameter” on page 219 parameter

installProfile parameter

The **installProfile** parameter can be used to specify the name of a java properties file used to specify installation settings for the editor.. This parameter is intended to be used by application writers who wish to alter the installation setting for one or more of the editor parameters.

Availability

query command

set command

Scope

Global.

Syntax

```
query installProfile
set installProfile [name]
```

Parameters

name

Use the *name* parameter to specify the name of the properties file that you wish to use to indicate the editor's installation settings. *name* should include the package qualifier but not the **.properties** extension.

Description

If you do not specify a properties file, then the editor will use **com.ibm.lpex.core.Install**. This will locate the **Install.properties** file which is located in the **com.ibm.lpex.core** package.

If you wish to provide your own properties file for editor installation settings you should set the **installProfile** parameter before creating any document views.

Any of the editor parameters that are available with the **install** parameter can have installation settings specified in the install profile properties file.

Use the qualified install parameter as the key and the desired setting as the corresponding value. For example:

```
install.block.defaultType=stream
install.commandLine=on
install.expandHide=on
install.expandTabs=on
install.findText.asis=off
install.findText.block=off
install.findText.columns=off
install.findText.emphasis=on
install.findText.endColumn=80
install.findText.mark=off
install.findText.regularExpression=off
install.findText.startColumn=1
install.findText.wrap=on
install.font="Monospaced-12"
install.hideSequenceNumbers=off
install.lineNumbers=off
install.messageLine=on
install.print.bottomMargin=25
```

```
install.print.font=screen  
install.print.leftMargin=10  
install.print.lineNumbers=off  
install.print.rightMargin=10  
install.print.tokenized=off  
install.print.topMargin=25  
install.save.trim=on  
install.statusLine=on  
install.tabs=1 every 8  
install.updateProfile.baseProfile=lpex  
install.updateProfile.parser=associated
```

Examples

```
query installProfile  
set installProfile com.ibm.lpex.samples.Test  
set installProfile
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“block.defaultType parameter” on page 67 parameter
“commandLine parameter” on page 72 parameter
current parameter
“default parameter” on page 83 parameter
“expandHide parameter” on page 90 parameter
“expandTabs parameter” on page 92 parameter
“findText.asis parameter” on page 96 parameter
“findText.block parameter” on page 98 parameter
“findText.columns parameter” on page 100 parameter
“findText.emphasis parameter” on page 102 parameter
“findText.endColumn parameter” on page 104 parameter
“findText.findText parameter” on page 106 parameter
“findText.mark parameter” on page 108 parameter
“findText.regularExpression parameter” on page 110 parameter
“findText.replaceText parameter” on page 112 parameter
“findText.startColumn parameter” on page 114 parameter
“findText.wrap parameter” on page 116 parameter
“font parameter” on page 118 parameter
“hideSequenceNumbers parameter” on page 125 parameter
“install parameter” on page 129 parameter
“lineNumbers parameter” on page 138 parameter
“messageLine parameter” on page 150 parameter
“popup parameter” on page 156 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
“save.textLimit parameter” on page 176 parameter
“save.trim parameter” on page 178 parameter
“sequenceNumbers parameter” on page 182 parameter
“statusLine parameter” on page 185 parameter
“tabs parameter” on page 189 parameter
“updateProfile.baseProfile parameter” on page 194 parameter
“updateProfile.extensions parameter” on page 196 parameter
“updateProfile.noParser parameter” on page 197 parameter
“updateProfile.palette parameter” on page 199 parameter
“updateProfile.paletteAttributes parameter” on page 201 parameter
“updateProfile.palettes parameter” on page 203 parameter
“updateProfile.parser parameter” on page 204 parameter
“updateProfile.parserAssociation parameter” on page 206 parameter
“updateProfile.parserClass parameter” on page 208 parameter
“updateProfile.parsers parameter” on page 210 parameter
“updateProfile.userActions parameter” on page 211 parameter
“updateProfile.userCommands parameter” on page 213 parameter
“updateProfile.userKeyActions parameter” on page 215 parameter
“updateProfile.userMouseActions parameter” on page 217 parameter
“updateProfile.userProfile parameter” on page 219 parameter

keyAction parameter

The **keyAction** parameter may be used to query or set an action assignment for a specified key.

Availability

query command

set command

Scope

The current view.

Syntax

```
query keyAction.[modifier-][...]key[.context][...][.secondary]  
set keyAction.[modifier-][...]key[.context][...][.secondary] [action]
```

Parameters

modifier

The *modifier* parameter can be any of **a**, **c**, **m**, or **s**. **a** is used to indicate the alt key modifier. **c** is used to indicate the ctrl key modifier. **m** is used to indicate the meta key modifier. **s** is used to indicate the shift key modifier. For example if you want to set a key for Ctrl+Alt+A you can specify "c-a-a".

key

The *key* parameter can be any of the following:

- **a** through **z**
- **0** through **9**
- **f1** through **f12**
- **numpad0** through **numpad9**
- **accept**
- **add**
- **alt**
- **backQuote**
- **backSlash**
- **backSpace**
- **cancel**
- **capsLock**
- **clear**
- **closeBracket**
- **comma**
- **control**
- **convert**
- **decimal**
- **divide**
- **down**
- **end**
- **enter**
- **equals**
- **escape**
- **final**
- **help**
- **home**
- **insert**
- **kana**
- **kanji**
- **left**
- **meta**
- **modechange**
- **multiply**
- **nonconvert**
- **openBracket**
- **pageDown**
- **pageUp**
- **pause**
- **period**
- **printscreen**
- **quote**
- **right**
- **scrollLock**
- **semicolon**
- **separator**
- **shift**
- **slash**
- **subtract**
- **tab**
- **up**

<i>context</i>	The <i>context</i> parameter can be any of t , p , or c . t is used to indicate that the key should be available when the cursor is in the text area. p is used to indicate that the key should be available when the cursor is in the prefix area. c is used to indicate that the key should be available when the cursor is on the command line. If you do not specify a context, then the context is assumed to be t .
<i>secondary</i>	Use the secondary parameter if you wish to assign a secondary key to an action. Only one primary key can be assigned to an action at a time but you can assign as many secondary keys to the same action as you want. The primary key that is assigned to an action is the one that is returned by the actionKey parameter.
<i>action</i>	Use the <i>action</i> parameter to specify the action that you want to assign to the specified key. If you do not specify an action, then the key will not have an action assigned to it.

Examples

```
query keyAction.c-f
set keyAction.a-backSpace.t.p.secondary undo
```

RELATED REFERENCES

“query command” on page 47 command
 “set command” on page 52 command
 “actionKey parameter” on page 64 parameter
 “actionKeyText parameter” on page 65
 parameter
 “keys parameter” parameter
 “updateProfile.userKeyActions parameter” on
 page 215 parameter

keys parameter

The **keys** parameter can be used to list all keys to which actions have been assigned.

Availability

query command

Scope

The current view.

Syntax

```
query keys
```

Description

The keys that are returned from the query have the same syntax as used by the **keyAction** parameter.

Examples

```
query keys
```

RELATED REFERENCES

“query command” on page 47 command
“actionKey parameter” on page 64 parameter
“keyAction parameter” on page 133
parameter

length parameter

The **length** parameter may be used to determine the length of the current element.

Availability

query command

Scope

The current element of the current view.

Syntax

query length

Examples

query length

RELATED REFERENCES

“query command” on page 47 command
“text parameter” on page 191 parameter

line parameter

The **line** parameter may be used to determine the line number of the current element. The value returned by the **line** parameter is usually the same as the value returned by the **element** parameter unless there are **show** elements in the document. Show elements are not counted when calculating the element’s line number.

Availability

query command

Scope

The current element of the current view.

Syntax

query line

Examples

query line

RELATED REFERENCES

“locate command” on page 42 command
“query command” on page 47 command
“element parameter” on page 86 parameter
“elements parameter” on page 88 parameter
“lines parameter” on page 139 parameter
“show parameter” on page 184 parameter

lineNumbers parameter

The **lineNumbers** parameter may be used to query or set the visibility of the line numbers. The line numbers are displayed in the prefix area to the left of the text area.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

lineNumbers is scoped to the current view.
current.lineNumbers is scoped to the current view.
default.lineNumbers is globally scoped.
install.lineNumbers is globally scoped.

Syntax

```
query lineNumbers
set lineNumbers [ default | on | off ]
query current.lineNumbers
query default.lineNumbers
set default.lineNumbers [ install | on | off ]
query install.lineNumbers
```

Parameters

default

If you specify the **default** parameter for the **set lineNumbers** command, then the current view will use the value of **default.lineNumbers** to determine the visibility of the line numbers.

on

If you specify the **on** parameter for the **set lineNumbers** command, then the line numbers will be visible on the current view. If you specify the **on** parameter for the **set default.lineNumbers** command, then the line numbers will be visible for all views that have **lineNumbers** set to **default**.

off

If you specify the **off** parameter for the **set lineNumbers** command, then the line numbers will not be visible on the current view. If you specify the **off** parameter for the **set default.lineNumbers** command, then the line numbers will not be visible for all views that have **lineNumbers** set to **default**. If you specify the **install** parameter for the **set default.lineNumbers** command, then all of the views that have **lineNumbers** set to **default** will use the value of **install.lineNumbers** to determine the visibility of the line numbers.

install

Description

If you do not specify any of the parameters for the **set lineNumbers** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.lineNumbers** command, then **install** is assumed.

The **query current.lineNumbers** command will return **on** if the line numbers are visible or **off** if the command line is not visible.

Examples

```
query lineNumbers
set lineNumbers off
query current.lineNumbers
query default.lineNumbers
set default.lineNumbers off
query install.lineNumbers
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“prefixAreaWidth parameter” on page 158 parameter

lines parameter

The **lines** parameter may be used to determine the total number of lines in the current document. The value returned by the **lines** parameter is the same as the value returned by the **elements** parameter unless the document contains **show** elements. Show elements are not counted by the **lines** parameter.

Availability

query command

Scope

The current view’s document.

Syntax

query lines

Examples

query lines

RELATED REFERENCES

“locate command” on page 42 command
“query command” on page 47 command
“element parameter” on page 86 parameter
“elements parameter” on page 88 parameter
“line parameter” on page 137 parameter
“show parameter” on page 184 parameter

maintainSequenceNumbers parameter

The **maintainSequenceNumbers** parameter may be used to query or set if the editor is maintaining the current document’s sequence numbers. Sequence numbers are defined with the **sequenceNumbers** parameter. When sequence numbers are being maintained, they are kept in sequential order. If, when **maintainSequenceNumbers** is set to on, the document’s sequence numbers are not in order, the **resequence** command will be issued and the document’s sequence numbers will be reordered starting at 100 and incrementing at intervals of 100.

Availability

query command
set command

Scope

The current view’s document.

Syntax

```
query maintainSequenceNumbers  
set maintainSequenceNumbers { on | off }
```

Parameters

on

If you specify the **on** parameter then the document’s sequence numbers will be maintained in sequential order.

off

If you specify the **off** parameter then the document’s sequence numbers will not be maintained.

Examples

```
query maintainSequenceNumbers  
set maintainSequenceNumbers on
```

RELATED REFERENCES

“query command” on page 47 command
“resequence command” on page 50 command
“set command” on page 52 command
“hideSequenceNumbers parameter” on
page 125 parameter
“sequenceNumber parameter” on page 180
parameter
“sequenceNumbers parameter” on page 182
parameter

mark parameter

The **mark** parameter can be used to query, set, or remove a named or unnamed mark. A mark is a label that has been assigned to a contiguous portion of the document. As the document is edited, the mark’s location is maintained such that it stays with the text that it started on. There are two types of marks, element marks and character marks. Element marks have no column positions and refer only to complete elements. Character marks have column positions and can refer to partial elements. You can also specify if a mark should be a sticky mark or not. A sticky mark will expand if text is added immediately before or immediately after the mark.

Availability

query command
set command

Scope

The current view.

Syntax

```
query mark.[name | #id]
set mark.[name | #id] { clear
                        [ sticky ] element [ element1 [ element2 ] ]
                        [ sticky ] [ element1 [ column1 [ element2 [ column2 ] ] ] ]
                        }
```

Parameters

name

Use the *name* parameter to specify a new name or the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.

id

Use the *id* parameter to specify the id of an existing mark.

clear

Use the **clear** parameter to remove the specified mark.

[sticky] element [*element1* [*element2*]]

Specify the optional **sticky** parameter if you want this mark to expand if elements are added right after or right before the mark. Use the **element** parameter to indicate that the mark should be created (or recreated) as an element mark. Element marks refer only to complete elements. Specify the optional parameters *element1* and *element2* to indicate the range of the element mark. *element1* and *element2* must be positive integers and refer to element ordinal numbers. If you do not specify *element2*, then the mark is created with *element1* as both the start and end of the mark. If you do not specify *element1* or *element2*, then the mark is created with the current element as both the start and end of the mark.

[sticky] [*element1* [*column1* [*element2* [*column2*]]]]

Specify the optional **sticky** parameter if you want this mark to expand if text is added immediately before or immediately after the mark. If you do not specify any parameters, then a character mark is created. A character mark has a starting element with a starting column and ending element with an ending column. By default the character mark is created with both its starting point and ending point at the current cursor position. If you only specify *element1*, then the mark starts and ends at column 1 of the specified element. If you only specify *element1* and *column1*, then the element starts and ends at the specified position. If you only specify *element1*, *column1*, and *element2*, then the mark starts at *column1* of *element1* and ends at column 1 of *element2*.

Description

If you do not specify a mark name or a mark id and you do not specify the **clear** option, then you will create an unnamed mark. You will only be able to refer to this unnamed mark by its id. You may determine the marks id by issuing "**query markid.**" (notice the trailing period). "**query markid.**" will return the id of the last mark created. If you do not specify a mark name or a mark id and you do specify the **clear** option, then you will remove the last mark created.

Examples

```
query mark.test
set mark.test element 5 10
set mark.test sticky 1 1 1 1
set mark.test clear
```

RELATED REFERENCES

“locate command” on page 42 command
“query command” on page 47 command
“set command” on page 52 command
“headerMark parameter” on page 121 parameter
“markExcluded parameter” parameter
“markExcludedHeader parameter” on page 144 parameter
“markHighlight parameter” on page 145 parameter
“markId parameter” on page 146 parameter
“markIncluded parameter” on page 147 parameter
“markProtect parameter” on page 148 parameter
“markStyle parameter” on page 149 parameter
findMark action
findQuickMark action
nameMark action
setQuickMark action

markExcluded parameter

The **markExcluded** parameter can be used to query or set the excluded attribute of a mark. If a mark is excluded from the current view, then none of the elements that are associated with the mark will be visible.

Availability

query command
set command

Scope

The specified mark in the current view.

Syntax

```
query markExcluded.[name | #id]  
set markExcluded.[name | #id] { on | off }
```

Parameters

name

Use the *name* parameter to specify the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.

id

Use the *id* parameter to specify the id of an existing mark.

on

Use the **on** parameter to indicate that the specified mark should be excluded from the current view.

off

Use the **off** parameter to indicate that the specified mark should not be excluded from the current view.

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markExcluded.test
set markExcluded.test on
set markExcluded.#3 off
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“headerMark parameter” on page 121

parameter

“mark parameter” on page 141 parameter

“markExcludedHeader parameter” parameter

“markId parameter” on page 146 parameter

“markIncluded parameter” on page 147

parameter

markExcludedHeader parameter

The **markExcludedHeader** parameter can be used to query or set if the specified mark should have a header element when it is excluded. A header element is a **show** element that replaces the elements that are associated with the mark when the mark is excluded. The header element contains text which indicates the number of lines that have been excluded.

Availability

query command

set command

Scope

The specified mark in the current view.

Syntax

```
query markExcludedHeader.[name | #id]
set markExcludedHeader.[name | #id] { on | off }
```

Parameters

name

Use the *name* parameter to specify the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.

id

Use the *id* parameter to specify the id of an existing mark.

on

Use the **on** parameter to indicate that the specified mark should be replaced by a mark header element when the mark is excluded from the current view.

off

Use the **off** parameter to indicate that the specified mark should not be replaced by a mark header element when the mark is excluded from the current view.

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markExcludedHeader.test
set markExcludedHeader.test on
set markExcludedHeader.#3 off
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“headerMark parameter” on page 121 parameter
“mark parameter” on page 141 parameter
“markExcluded parameter” on page 143 parameter
“markId parameter” on page 146 parameter
“markIncluded parameter” on page 147 parameter

markHighlight parameter

The **markHighlight** parameter can be used to query or set if the specified mark should be highlighted. If a mark is highlighted, then it will be drawn with the style attributes indicated by the **markStyle** parameter.

Availability

query command
set command

Scope

The specified mark in the current view.

Syntax

```
query markHighlight.[name | #id]  
set markHighlight.[name | #id] { on | off }
```

Parameters

name

Use the *name* parameter to specify the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.

id

Use the *id* parameter to specify the id of an existing mark.

on

Use the **on** parameter to indicate that the specified mark should be highlighted.

off

Use the **off** parameter to indicate that the specified mark should not be highlighted.

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markHighlight.test
set markHighlight.test on
set markHighlight.#3 off
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“mark parameter” on page 141 parameter
“markId parameter” parameter
“markStyle parameter” on page 149
parameter

markId parameter

The **markId** parameter can be used to query the id number of the specified mark. This is usually used to determine the id of an unnamed mark immediately after it has been created.

Availability

query command

Scope

The specified mark in the current view.

Syntax

```
query markId.[name | #id]
```

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markId.
query markId.test
query markId.#3
```


RELATED REFERENCES

“query command” on page 47 command
“mark parameter” on page 141 parameter
“markExcluded parameter” on page 143 parameter
“markExcludedHeader parameter” on page 144 parameter
“markHighlight parameter” on page 145 parameter
“markIncluded parameter” parameter
“markProtect parameter” on page 148 parameter
“markStyle parameter” on page 149 parameter

markIncluded parameter

The **markIncluded** parameter can be used to query or set the included attribute of a mark. If a mark is included in the current view, then only the elements that are associated with the specified element (and any other included marks) will be visible.

Availability

query command
set command

Scope

The specified mark in the current view.

Syntax

```
query markIncluded.[name | #id]  
set markIncluded.[name | #id] { on | off }
```

Parameters

name

Use the *name* parameter to specify the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.

id

Use the *id* parameter to specify the id of an existing mark.

on

Use the **on** parameter to indicate that the specified mark should be included in the current view.

off

Use the **off** parameter to indicate that the specified mark should not be included in the current view.

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markIncluded.test
set markIncluded.test on
set markIncluded.#3 off
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“headerMark parameter” on page 121 parameter
“mark parameter” on page 141 parameter
“markExcluded parameter” on page 143 parameter
“markId parameter” on page 146 parameter

markProtect parameter

The **markProtect** parameter can be used to query or set the protect attribute of a mark. If a mark is protected, then you will not be able to delete or change any of the elements associated with the mark. You will also be unable to insert new non **show** elements between two elements contained within the protected mark.

Availability

query command
set command

Scope

The specified mark in the current view.

Syntax

```
query markProtect.[name | #id]  
set markProtect.[name | #id] { on | off }
```

Parameters

<i>name</i>	Use the <i>name</i> parameter to specify the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.
<i>id</i>	Use the <i>id</i> parameter to specify the id of an existing mark.
on	Use the on parameter to indicate that the specified mark should be protected.
off	Use the off parameter to indicate that the specified mark should not be protected.

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markProtect.test
set markProtect.test on
set markProtect.#3 off
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“mark parameter” on page 141 parameter
“markId parameter” on page 146 parameter

markStyle parameter

The **markStyle** parameter can be used to query or set the style character that is associated with a mark. The style character is used to determine the style attributes of the mark that should be used to draw the text associated with the mark if the **markHighlight** parameter is **on** for the specified mark.

Availability

query command
set command

Scope

The specified mark in the current view.

Syntax

```
query markStyle.[name | #id]  
set markStyle.[name | #id] c
```

Parameters

name

Use the *name* parameter to specify the name of an existing mark. Names may contain any characters but spaces and may not start with the # character.

id

Use the *id* parameter to specify the id of an existing mark.

c

Use the *c* parameter to specify the style character.

Description

If you do not specify a mark name or a mark id, then the last mark created will be used.

Examples

```
query markStyle.test  
set markStyle.test m  
set markStyle.#3 m
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“mark parameter” on page 141 parameter
“markHighlight parameter” on page 145 parameter
“markId parameter” on page 146 parameter
“styleAttributes parameter” on page 187

messageLine parameter

The **messageLine** parameter may be used to query or set the visibility of the message line.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

messageLine is scoped to the current view.
current.messageLine is scoped to the current view.
default.messageLine is globally scoped.
install.messageLine is globally scoped.

Syntax

```
query messageLine
set messageLine [ default | on | off ]
query current.messageLine
query default.messageLine
set default.messageLine [ install | on | off ]
query install.messageLine
```

Parameters

default

If you specify the **default** parameter for the **set messageLine** command, then the current view will use the value of **default.messageLine** to determine the visibility of the message line.

on

If you specify the **on** parameter for the **set messageLine** command, then the message line will be visible on the current view. If you specify the **on** parameter for the **set default.messageLine** command, then the message line will be visible for all views that have **messageLine** set to **default**.

off

If you specify the **off** parameter for the **set messageLine** command, then the message line will not be visible on the current view. If you specify the **off** parameter for the **set default.messageLine** command, then the message line will not be visible for all views that have **messageLine** set to **default**.

install

If you specify the **install** parameter for the **set default.messageLine** command, then all of the views that have **messageLine** set to **default** will use the value of **install.messageLine** to determine the visibility of the message line.

Examples

```
query messageLine
set messageLine off
query current.messageLine
query default.messageLine
set default.messageLine off
query install.messageLine
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“messageText parameter” parameter

messageText parameter

The **messageText** parameter may be used to query or set the text that is displayed in the message line.

Availability

query command
set command

Scope

The current view.

Syntax

```
query messageText
set messageText text
```

Parameters

text

Use the *text* parameter to specify the text that should be displayed in the message line.

Examples

```
query messageText
set messageText Good afternoon!
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“messageLine parameter” on page 150 parameter

mouseAction parameter

The **mouseAction** parameter may be used to query or set an action assignment for a specified mouse event.

Availability

query command

set command

Scope

The current view.

Syntax

```
query mouseAction.[modifier-][...]{clicked.count |
                                   dragged |
                                   entered |
                                   exited |
                                   moved |
                                   popup |
                                   pressed.count |
                                   released.count
                                   }[.context][...]
set mouseAction.[modifier-][...]{clicked.count |
                                   dragged |
                                   entered |
                                   exited |
                                   moved |
                                   popup |
                                   pressed.count |
                                   released.count
                                   }[.context][...] [action]
```

Parameters

modifier

The **modifier** parameter can be any of **1**, **2**, **3**, **a**, **c**, **m**, or **s**. **1** is used to indicate mouse button 1. **2** is used to indicate mouse button 2. **3** is used to indicate mouse button 3. **a** is used to indicate the alt key modifier. **c** is used to indicate the ctrl key modifier. **m** is used to indicate the meta key modifier. **s** is used to indicate the shift key modifier. For example if you want to specify an action for double clicking mouse button 1 with the Ctrl and Alt keys pressed, you can specify **c-a-1-clicked.2**.

clicked.count

Use the **clicked** parameter when to assign an action to a mouse clicked event. The *count* parameter indicates the number of button clicks that are associated with the event.

dragged

Use the **dragged** parameter to assign an action to the mouse dragged event.

entered

Use the **entered** parameter to assign an action to the mouse entered event.

exited

Use the **exited** parameter to assign an action to the mouse exited event.

moved

Use the **moved** parameter to assign an action to the mouse moved event.

popup

Use the **popup** parameter to assign an action to the pop-up trigger mouse event.

pressed.count

Use the **pressed** parameter to assign an action to a mouse pressed event. The *count* parameter indicates the number of button clicks that are associated with the event.

`released.count`

`context`

`action`

Use the **released** parameter to assign an action to a mouse released event. The *count* parameter indicates the number of button clicks that are associated with the event.

The *context* parameter can be any of **t**, **p**, or **e**. **t** is used to indicate that the mouse action should be available when the mouse cursor is in the text area. **p** is used to indicate that the mouse action should be available when the mouse cursor is in the prefix area. **e** is used to indicate that the mouse action should be available when the mouse cursor is in the expand/hide area. If you do not specify a context, then the context is assumed to be **t**.

Use the *action* parameter to specify the action that you want assigned to the specified mouse event. If you do not specify an action, then the event will have no action assigned to it.

Examples

```
query mouseAction.popup
set mouseAction.1-pressed.1.t.p cursorToMouse
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“mouseEvents parameter” parameter

“updateProfile.userMouseActions parameter”
on page 217 parameter

mouseEvents parameter

The **mouseEvents** parameter can be used to list all mouse events to which actions have been assigned.

Availability

query command

Scope

The current view.

Syntax

```
query mouseEvents
```

Description

The mouse events that are returned from the query have the same syntax as used by the **mouseAction** parameter.

Examples

```
query mouseEvents
```

RELATED REFERENCES

“query command” on page 47 command
“mouseAction parameter” on page 151
parameter

name parameter

The **name** parameter may be used to query or set the document’s file name. The file name is used by the **save** and **load** commands.

Availability

query command
set command

Scope

The current view’s document.

Syntax

```
query name  
set name fileName
```

Parameters

fileName

Use the *fileName* parameter to specify the desired file name.

Examples

```
query name  
set name test.java
```

RELATED REFERENCES

“load command” on page 42 command
“query command” on page 47 command
“save command” on page 50 command
“set command” on page 52 command

palette parameter

The **palette** parameter can be used to determine the color palette that was used the last time the **updateProfile** command was issued. The **updateProfile** command is normally issued as part of opening a file. The color palette is used to determine the colors that are used by the editor. Use the **current.updateProfile.palettes** parameter to determine the available palettes.

Availability

query command

Scope

The current view.

Syntax

query palette

Examples

query palette

RELATED REFERENCES

“query command” on page 47 command
“updateProfile command” on page 55
command
“updateProfile.palette parameter” on
page 199 parameter
“updateProfile.paletteAttributes parameter”
on page 201 parameter
“updateProfile.palettes parameter” on
page 203 parameter

parser parameter

The **parser** parameter may be used to determine the name of the parser used by the current view.

Availability

query command

Scope

The current view.

Syntax

query parser

Examples

query parser

RELATED REFERENCES

“query command” on page 47 command
“updateProfile.noParser parameter” on
page 197 parameter
“updateProfile.parserAssociation parameter”
on page 206 parameter
“updateProfile.parserClass parameter” on
page 208 parameter
“updateProfile.parser parameter” on page 204
parameter

pixelPosition parameter

The **pixelPosition** parameter may be used to determine the pixel offset of the current cursor position. The pixel offset is calculated from the beginning of the element’s text. The prefix area, the expand/hide area and the current horizontal scroll are will not affect the this calculation.

Availability

query command

Scope

The current view.

Syntax

```
query pixelPosition
```

Examples

```
query pixelPosition
```

RELATED REFERENCES

“query command” on page 47 command

“displayPosition parameter” on page 85

parameter

“position parameter” on page 158 parameter

“scroll parameter” on page 179 parameter

“textAreaWidth parameter” on page 191

parameter

“textWidth parameter” on page 192

parameter

popup parameter

The **popup** parameter may be used to query or set the pop-up menu.

Availability

query command

set command

current parameter

default parameter

install parameter

Scope

popup is scoped to the current view.

current.popup is scoped to the current view.

default.popup is globally scoped.

install.popup is globally scoped.

Syntax

```
query popup
set popup { default
    | [ menuItemmenuAction
        | beginSubmenu menuItem
        | endSubmenu
        | separator
    ] [...]
}

query current.popup
query default.popup
set default.popup { install
    | [ menuItem menuAction
```

```

| beginSubmenu menuItem
| endSubmenu
| separator
| [...]
}
query install.popup

```

Parameters

default

```

[ menuItem menuAction
| beginSubmenu menuItem
| endSubmenu
| separator
| [...]

```

If you specify the **default** parameter for the **set popup** command, then the current view will use the value of **default.popup** as the pop-up menu for the current document view. If you specify a pop-up menu for the **set popup** command, then the current view will use the specified pop-up menu as its pop-up menu. If you specify a pop-up menu for the **set default.popup** command, then all document views that have **default.popup** set to **default** will use the specified pop-up menu.

Use the *menuItem* parameter to specify the text for the menu item. It may either be a key for a string in the **Resource.properties** file or a quoted string. The *menuAction* parameter must be an editor action.

Use the **beginSubmenu** parameter to start a new submenu. The *menuItem* parameter may be used to set the text for the new submenu. It may either be a key for a string in the **Resource.properties** file or a quoted string.

Use the **endSubmenu** parameter to terminate the current submenu.

Use the **separator** parameter to insert a separator between two menu items.

If you specify the **install** parameter for the **set default.popup** command, then all of the views that have **popup** set to **default** will use the value of **install.popup** as the pop-up menu.

install

Description

The **query current.popup** command will return the pop-up menu that is being used by the current view.

Examples

```

query popup
set popup "Cut" cut "Copy" copy "Paste" paste
query current.popup
query default.popup
set default.popup install
query install.popup

```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter

position parameter

The **position** parameter may be used to query or set the cursor’s column position on the current element.

Availability

query command
set command

Scope

The current view.

Syntax

```
query position  
set position n
```

Parameters

n

Use the *n* parameter to indicate the column position of the cursor.

Examples

```
query position  
set position 20
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“displayPosition parameter” on page 85 parameter

prefixAreaWidth parameter

The **prefixAreaWidth** parameter may be used to determine the width in pixels of the prefix area. The prefix area is the area to the left of the text area that is visible when **current.lineNumbers** returns on.

Availability

query command

Scope

The current view.

Syntax

query prefixAreaWidth

Examples

query prefixAreaWidth

RELATED REFERENCES

“query command” on page 47 command

“lineNumbers parameter” on page 138

parameter

“prefixPosition parameter” parameter

“prefixText parameter” on page 161

parameter

prefixPosition parameter

The **prefixPosition** parameter may be used to query or set the cursor’s column position in the prefix area. The prefix area is the area to the left of the text area that is visible when **current.lineNumbers** is **on**.

Availability

query command

set command

Scope

The current view.

Syntax

query prefixPosition

set prefixPosition *n*

Parameters

n

Use the *n* parameter to indicate the column number of the cursor.

Description

Note that unlike the **position** parameter, which is available for setting the column position within the text area, the **prefixPosition** parameter cannot be set to a position which is beyond the current length of the prefix text. You can determine the current prefix text with the **prefixText** parameter.

Examples

query prefixPosition

set prefixPosition 20

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“inPrefix parameter” on page 126 parameter
“lineNumbers parameter” on page 138 parameter
“prefixAreaWidth parameter” on page 158 parameter
“prefixText parameter” on page 161 parameter

prefixProtect parameter

The **prefixProtect** parameter may be used to query or set if the the cursor can be moved into the prefix area.

Availability

query command
set command

Scope

The current view.

Syntax

```
query prefixProtect  
set prefixProtect { on | off }
```

Parameters

on

If you specify the **on** parameter then you will not be able to move the cursor into the prefix area.

off

If you specify the **off** parameter then you will be able to move the cursor into the prefix area.

Description

The prefix area is the an area to the left of the text area that is visible when the **current.lineNumbers** parameter is **on**. If **prefixProtect** is set to **on**, then you will not be able to move the cursor into the prefix area. That is, you will not be able to set **inPrefix** to **on**.

Examples

```
query prefixProtect  
set prefixProtect on
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“processPrefix command” on page 46 command
“updateProfile command” on page 55 command
“baseProfile parameter” on page 66 parameter
“inPrefix parameter” on page 126 parameter
“lineNumbers parameter” on page 138 parameter
“prefixAreaWidth parameter” on page 158 parameter
“prefixPosition parameter” on page 159 parameter
“prefixText parameter” parameter
“updateProfile.baseProfile parameter” on page 194 parameter
prefixBackSpace action
prefixDelete action
prefixEnd action
prefixHome action
prefixLeft action
prefixRight action
prefixTruncate action
processPrefix action

prefixText parameter

The **prefixText** parameter may be used to query or set the current element’s prefix text. The prefix area is the area to the left of the text area that is visible when **current.lineNumbers** is **on**. The prefix text is used by the **processPrefix** command to implement prefix commands for base profiles like **ispf**, **seu**, and **xedit**.

Availability

query command
set command

Scope

The current element of the current view.

Syntax

```
query prefixText
set prefixText text
```

Parameters

text

Use the *text* parameter to specify the prefix text for the current element.

Examples

```
query prefixText
set prefixText cc
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“inPrefix parameter” on page 126 parameter
“lineNumbers parameter” on page 138 parameter
“prefixAreaWidth parameter” on page 158 parameter
“prefixPosition parameter” on page 159 parameter

print.bottomMargin parameter

The **print.bottomMargin** parameter may be used to query or set the bottom margin that will be used by the **print** command. The bottom margin is calculated in pixels.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

print.bottomMargin is scoped to the current view.
current.print.bottomMargin is scoped to the current view.
default.print.bottomMargin is globally scoped.
install.print.bottomMargin is globally scoped.

Syntax

```
query print.bottomMargin
set print.bottomMargin [ default | n ]
query current.print.bottomMargin
query default.print.bottomMargin
set default.print.bottomMargin [ install | n ]
query install.print.bottomMargin
```

Parameters

default

If you specify the **default** parameter for the **set print.bottomMargin** command, then the current view will use the value of **default.print.bottomMargin** for the **print** command.

n

If you specify the ***n*** parameter for the **set print.bottomMargin** command, then within the current view, the **print** command will use ***n*** as the bottom margin. If you specify the ***n*** parameter for the **set default.print.bottomMargin** command, then within all document views that have the **print.bottomMargin** set to **default**, the **print** command will use ***n*** as the bottom margin.

install

If you specify the **install** parameter for the **set default.print.bottomMargin** command, then all of the views that have **print.bottomMargin** set to **default** will use the value of **install.print.bottomMargin** to determine if the bottom margin that should be used by the **print** command.

Description

If you do not specify any of the parameters for the **set print.bottomMargin** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.bottomMargin** command, then **install** is assumed.

The **query current.print.bottomMargin** command will return the bottom margin that will be used, within the current view, by the **print** command.

Examples

```
query print.bottomMargin
set print.bottomMargin 20
query current.print.bottomMargin
query default.print.bottomMargin
set default.print.bottomMargin 15
query install.print.bottomMargin
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“print.font parameter” parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
print action

print.font parameter

The **print.font** parameter may be used to query or set the font that will be used by the **print** command.

Availability

query command
set command

current parameter
default parameter
install parameter

Scope

print.font is scoped to the current view.
current.print.font is scoped to the current view.
default.print.font is globally scoped.
install.print.font is globally scoped.

Syntax

```
query print.font
set print.font [ screen | default | "fontName" ]
query current.print.font
query default.print.font
set default.print.font [ screen | install | "fontName" ]
query install.print.font
```

Parameters

screen

If you specify the **screen** parameter for the **set print.font** command, then the current view will use the value of **current.font** as the font for the **print** command. If you specify the **screen** parameter for the **set default.print.font** command, then within all document views that have **print.font** set to **default**, the **print** command will use the value of **current.font** as the font for the **print** command.

default

If you specify the **default** parameter for the **set print.font** command, then the current view will use the value of **default.print.font** as the font for the **print** command.

fontName

If you specify the *fontName* parameter for the **set print.font** command, then within the current view, the **print** command will use the font indicated by **fontName**. If you specify the *fontName* parameter for the **set default.print.font** command, then within all document views that have the **print.font** set to **default**, the **print** command will use the font indicated by *fontName*. The *fontName* parameter should be in one of the following forms:

- *fontname-style-pointsize*
- *fontname-pointsize*
- *fontname-style*
- *fontname*

where *style* is one of **bold**, **bolditalic**, or **italic** and *pointsize* is a decimal representation of the point size.

install

If you specify the **install** parameter for the **set default.print.font** command, then all of the views that have **print.font** set to **default** will use the value of **install.print.font** to determine if the font that should be used by the **print** command.

Description

If you do not specify any of the parameters for the **set print.font** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.font** command, then **install** is assumed.

The **query current.print.font** command will return the font that will be used, within the current view, by the **print** command.

Examples

```
query print.font
set print.font screen
query current.print.font
query default.print.font
set default.print.font "Monospaced-14"
query install.print.font
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“font parameter” on page 118 parameter
“install parameter” on page 129 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.leftMargin parameter” parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
print action

print.leftMargin parameter

The **print.leftMargin** parameter may be used to query or set the left margin that will be used by the **print** command. The left margin is calculated in pixels.

Availability

query command
set command

current parameter
default parameter
install parameter

Scope

print.leftMargin is scoped to the current view.
current.print.leftMargin is scoped to the current view.
default.print.leftMargin is globally scoped.
install.print.leftMargin is globally scoped.

Syntax

```
query print.leftMargin
set print.leftMargin [ default | n ]
query current.print.leftMargin
query default.print.leftMargin
set default.print.leftMargin [ install | n ]
query install.print.leftMargin
```

Parameters

default

If you specify the **default** parameter for the **set print.leftMargin** command, then the current view will use the value of **default.print.leftMargin** for the **print** command.

n

If you specify the *n* parameter for the **set print.leftMargin** command, then within the current view, the **print** command will use *n* as the left margin. If you specify the *n* parameter for the **set default.print.leftMargin** command, then within all document views that have the **print.leftMargin** set to **default**, the **print** command will use *n* as the left margin.

install

If you specify the **install** parameter for the **set default.print.leftMargin** command, then all of the views that have **print.leftMargin** set to **default** will use the value of **install.print.leftMargin** to determine if the left margin that should be used by the **print** command.

Description

If you do not specify any of the parameters for the **set print.leftMargin** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.leftMargin** command, then **install** is assumed.

The **query current.print.leftMargin** command will return the left margin that will be used, within the current view, by the **print** command.

Examples

```
query print.leftMargin
set print.leftMargin 20
query current.print.leftMargin
query default.print.leftMargin
set default.print.leftMargin 15
query install.print.leftMargin
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.lineNumbers parameter” parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
print action

print.lineNumbers parameter

The **print.lineNumbers** parameter may be used to query or set if line numbers should be printed at the start of each line by the **print** command.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

print.lineNumbers is scoped to the current view.
current.print.lineNumbers is scoped to the current view.
default.print.lineNumbers is globally scoped.
install.print.lineNumbers is globally scoped.

Syntax

```
query print.lineNumbers
set print.lineNumbers [ default | on | off ]
query current.print.lineNumbers
query default.print.lineNumbers
set default.print.lineNumbers [ install | on | off ]
query install.print.lineNumbers
```

Parameters

default

If you specify the **default** parameter for the **set print.lineNumbers** command, then the current view will use the value of **default.print.lineNumbers** to determine if line numbers should be printed by the **print** command.

on

If you specify the **on** parameter for the **set print.lineNumbers** command, then the **print** command will print line numbers for the current view. If you specify the **on** parameter for the **set default.print.lineNumbers** command, then the **print** command will print line numbers for all of document views that have **print.lineNumbers** set to **default**.

off

If you specify the **off** parameter for the **set print.lineNumbers** command, then the **print** command not print line numbers for the current view. If you specify the **off** parameter for the **set default.print.lineNumbers** command, then the **print** command will not print line numbers for all of document views that have **print.lineNumbers** set to **default**.

install

If you specify the **install** parameter for the **set default.print.lineNumbers** command, then all of the views that have **print.lineNumbers** set to **default** will use the value of **install.print.lineNumbers** to determine if the **print** command should print line numbers or not.

Description

If you do not specify any of the parameters for the **set print.lineNumbers** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.lineNumbers** command, then **install** is assumed.

The **query current.print.lineNumbers** command will return **on** if, for the current view, the **print** command will print line numbers and **off** if, for the current view, the **print** command will not print line numbers.

Examples

```
query print.lineNumbers
set print.lineNumbers off
query current.print.lineNumbers
query default.print.lineNumbers
set default.print.lineNumbers on
query install.print.lineNumbers
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.rightMargin parameter” parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
print action

print.rightMargin parameter

The **print.rightMargin** parameter may be used to query or set the right margin that will be used by the **print** command. The right margin is calculated in pixels.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

print.rightMargin is scoped to the current view.
current.print.rightMargin is scoped to the current view.
default.print.rightMargin is globally scoped.
install.print.rightMargin is globally scoped.

Syntax

```
query print.rightMargin
set print.rightMargin [ default | n ]
query current.print.rightMargin
query default.print.rightMargin
set default.print.rightMargin [ install | n ]
query install.print.rightMargin
```

Parameters

default

If you specify the **default** parameter for the **set print.rightMargin** command, then the current view will use the value of **default.print.rightMargin** for the **print** command.

n

If you specify the *n* parameter for the **set print.rightMargin** command, then within the current view, the **print** command will use *n* as the right margin. If you specify the *n* parameter for the set

default.print.rightMargin command, then within all document views that have the **print.rightMargin** set to **default**, the **print** command will use *n* as the right margin.

install

If you specify the **install** parameter for the **set default.print.rightMargin** command, then all of the views that have **print.rightMargin** set to **default** will use the value of **install.print.rightMargin** to determine if the right margin that should be used by the **print** command.

Description

If you do not specify any of the parameters for the **set print.rightMargin** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.rightMargin** command, then **install** is assumed.

The **query current.print.rightMargin** command will return the right margin that will be used, within the current view, by the **print** command.

Examples

```
query print.rightMargin
set print.rightMargin 20
query current.print.rightMargin
query default.print.rightMargin
set default.print.rightMargin 15
query install.print.rightMargin
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.tokenized parameter” on page 171 parameter
“print.topMargin parameter” on page 172 parameter
print action

print.tokenized parameter

The **print.tokenized** parameter may be used to query or set if the **print** command should use the current style attributes to print the document text.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

print.tokenized is scoped to the current view.
current.print.tokenized is scoped to the current view.
default.print.tokenized is globally scoped.
install.print.tokenized is globally scoped.

Syntax

```
query print.tokenized
set print.tokenized [ default | on | off ]
query current.print.tokenized
query default.print.tokenized
set default.print.tokenized [ install | on | off ]
query install.print.tokenized
```

Parameters

default

If you specify the **default** parameter for the **set print.tokenized** command, then the current view will use the value of **default.print.tokenized** to determine if the **print** command should use the current style attributes to print the document's text.

on

If you specify the **on** parameter for the **set print.tokenized** command, then the **print** command will use the current style attributes to print the current view's document text. If you specify the **on** parameter for the **set default.print.tokenized** command, then the **print** command will use the view's current style attributes to print for all the document views that have **print.tokenized** set to **default**.

off

If you specify the **off** parameter for the **set print.tokenized** command, then the **print** command will not use the current style attributes to print the current view's document text. If you specify the **off** parameter for the **set default.print.tokenized** command, then the **print** command will not use the view's current style attributes to print for all the document views that have **print.tokenized** set to **default**.

install

If you specify the **install** parameter for the **set default.print.tokenized** command, then all of the views that have **print.tokenized** set to **default** will use the value of **install.print.tokenized** to determine if the **print** command should use the view's style attributes to print the document text.

Description

If you do not specify any of the parameters for the **set print.tokenized** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.tokenized** command, then **install** is assumed.

The **query current.print.tokenized** command will return **on** if, for the current view, the **print** command will use the view's style attributes to print the document text and **off** if, for the current view, the **print** command will not use the view's style attributes to print the document text.

Examples

```
query print.tokenized
set print.tokenized off
query current.print.tokenized
query default.print.tokenized
set default.print.tokenized on
query install.print.tokenized
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.topMargin parameter” parameter
print action

print.topMargin parameter

The **print.topMargin** parameter may be used to query or set the top margin that will be used by the **print** command. The top margin is calculated in pixels.

Availability

query command
set command

current parameter
default parameter
install parameter

Scope

print.topMargin is scoped to the current view.
current.print.topMargin is scoped to the current view.
default.print.topMargin is globally scoped.
install.print.topMargin is globally scoped.

Syntax

```
query print.topMargin
set print.topMargin [ default | n ]
query current.print.topMargin
query default.print.topMargin
set default.print.topMargin [ install | n ]
query install.print.topMargin
```

Parameters

default

If you specify the **default** parameter for the **set print.topMargin** command, then the current view will use the value of **default.print.topMargin** for the **print** command.

n

If you specify the ***n*** parameter for the **set print.topMargin** command, then within the current view, the **print** command will use ***n*** as the top margin. If you specify the ***n*** parameter for the **set default.print.topMargin** command, then within all document views that have the **print.topMargin** set to **default**, the **print** command will use ***n*** as the top margin.

install

If you specify the **install** parameter for the **set default.print.topMargin** command, then all of the views that have **print.topMargin** set to **default** will use the value of **install.print.topMargin** to determine if the top margin that should be used by the **print** command.

Description

If you do not specify any of the parameters for the **set print.topMargin** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.print.topMargin** command, then **install** is assumed.

The **query current.print.topMargin** command will return the top margin that will be used, within the current view, by the **print** command.

Examples

```
query print.topMargin
set print.topMargin 20
query current.print.topMargin
query default.print.topMargin
set default.print.topMargin 15
query install.print.topMargin
```

RELATED REFERENCES

“print command” on page 44 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“print.bottomMargin parameter” on page 162 parameter
“print.font parameter” on page 163 parameter
“print.leftMargin parameter” on page 165 parameter
“print.lineNumbers parameter” on page 167 parameter
“print.rightMargin parameter” on page 169 parameter
“print.tokenized parameter” on page 171 parameter
print action

readonly parameter

The **readonly** parameter may be used to query or set if the document can be edited with the current view.

Availability

query command
set command

Scope

The current view.

Syntax

query readonly
set readonly { on | off }

Parameters

on

Use the **on** parameter to indicate that the document should be protected from changes through the current document view.

off

Use the **off** parameter to indicate that the document may be changed through the current document view.

Examples

```
query readonly  
set readonly on
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command

recording parameter

The **recording** parameter may be used to query or set if the editor is recording document changes. Recorded document changes may be undone by the **undo** command.

Availability

query command
set command

Scope

The current view's document.

Syntax

```
query recording
set recording { on | off }
```

Parameters

on

If you specify the **on** parameter then changes to the document will be recorded.

off

If you specify the **off** parameter then changes to the document will not be recorded.

Examples

```
query recording
set recording on
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“undo command” on page 54 command
“changes parameter” on page 69 parameter
“dirty parameter” on page 85 parameter
undo action
redo action

rowHeight parameter

The **rowHeight** parameter may be used to determine the height in pixels of the current row of text.

Availability

query command

Scope

The current view.

Syntax

```
query rowHeight
```

Examples

`query rowHeight`

RELATED REFERENCES

“query command” on page 47 command

“cursorRow parameter” on page 82
parameter

“rows parameter” parameter

rows parameter

The **rows** parameter may be used to determine the number of rows of text that can currently be displayed in the text window.

Availability

query command

Scope

The current view.

Syntax

`query rows`

Examples

`query rows`

RELATED REFERENCES

“query command” on page 47 command

“cursorRow parameter” on page 82

parameter

“rowHeight parameter” on page 175

parameter

save.textLimit parameter

The **save.textLimit** parameter may be used to query or set the maximum line length used by the **save** command.

Availability

query command

set command

current parameter

default parameter

install parameter

Scope

save.textLimit is scoped to the current document.

current.save.textLimit is scoped to the current document.

default.save.textLimit is globally scoped.

install.save.textLimit is globally scoped.

Syntax

```
query save.textLimit
set save.textLimit [ default | n ]
query current.save.textLimit
query default.save.textLimit
set default.save.textLimit [ install | n ]
query install.save.textLimit
```

Parameters

default

If you specify the **default** parameter for the **set save.textLimit** command, then the current document will use the value of **default.save.textLimit** for the **save** command.

n

If you specify the *n* parameter for the **set save.textLimit** command, then within the current document, the **save** command will use *n* as the text limit. If you specify the *n* parameter for the **set default.save.textLimit** command, then within all documents that have the **save.textLimit** parameter set to **default**, the **save** command will use *n* as the text limit.

install

If you specify the **install** parameter for the **set default.save.textLimit** command, then all of the documents that have **save.textLimit** set to **default** will use the value of **install.save.textLimit** to determine if the text limit that should be used by the **save** command.

Description

If you do not specify any of the parameters for the **set save.textLimit** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.save.textLimit** command, then **install** is assumed.

The **query current.save.textLimit** command will return the text limit that will be used, within the current document, by the **save** command.

Examples

```
query save.textLimit
set save.textLimit 80
query current.save.textLimit
query default.save.textLimit
set default.save.textLimit 85
query install.save.textLimit
```

RELATED REFERENCES

“query command” on page 47 command
“save command” on page 50 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“length parameter” on page 137 parameter
save action

save.trim parameter

The **save.trim** parameter may be used to query or set if the **save** command should trim trailing blanks from all of the lines in the document.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

save.trim is scoped to the current document.
current.save.trim is scoped to the current document.
default.save.trim is globally scoped.
install.save.trim is globally scoped.

Syntax

```
query save.trim
set save.trim [ default | on | off ]
query current.save.trim
query default.save.trim
set default.save.trim [ install | on | off ]
query install.save.trim
```

Parameters

default

If you specify the **default** parameter for the **set save.trim** command, then the current view will use the value of **default.save.trim** to determine if the **save** command should trim trailing blanks from all the lines in the document.

on

If you specify the **on** parameter for the **set save.trim** command, then the **save** command will trim trailing blanks from all the lines in the document. If you specify the **on** parameter for the set **default.save.trim** command, then the **save** command will trim trailing blanks from all the lines in the document for all of document views that have **save.trim** set to **default**.

off

If you specify the **off** parameter for the **set save.trim** command, then the **save** command will not trim trailing blanks from all the lines in the document. If you specify the **off** parameter for the **set default.save.trim** command, then the **save** command will not trim trailing blanks from all the lines in the document for all of document views that have **save.trim** set to **default**.

install

If you specify the **install** parameter for the **set default.save.trim** command, then all of the views that have **save.trim** set to **default** will use the value of **install.save.trim** to determine if the **save** command trim trailing blanks from all the lines in the document or not.

Description

If you do not specify any of the parameters for the **set save.trim** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.save.trim** command, then **install** is assumed.

The **query current.save.trim** command will return **on** if, for the current view, the **save** command will trim trailing blanks and **off** if, for the current view, the **save** command will not trim trailing blanks.

Examples

```
query save.trim
set save.trim off
query current.save.trim
query default.save.trim
set default.save.trim on
query install.save.trim
```

RELATED REFERENCES

“save command” on page 50 command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
save action

scroll parameter

The **scroll** parameter may be used to query or set the number of pixels that the current view is scrolled to the right.

Availability

query command
set command

Scope

The current view.

Syntax

```
query scroll  
set scroll n
```

Parameters

n

Use the *n* parameter to indicate the number of pixels that you want the current view to be scrolled to the right.

Description

Note that the scroll value that you indicate will be adjusted so that the cursor remains visible.

Examples

```
query scroll  
set scroll 20
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“displayPosition parameter” on page 85 parameter

“pixelPosition parameter” on page 155 parameter

“position parameter” on page 158 parameter

“textAreaWidth parameter” on page 191 parameter

“textWidth parameter” on page 192 parameter

sequenceNumber parameter

The **sequenceNumber** parameter may be used to query or set the current element’s sequence number.

Availability

```
query command  
set command
```

Scope

The current element of the current view.

Syntax

```
query sequenceNumber  
set sequenceNumber n
```

Parameters

n

Use the *n* parameter to specify the element's sequence number.

Examples

```
query sequenceNumber
set sequenceNumber 100
```

RELATED REFERENCES

“query command” on page 47 command
“resequence command” on page 50 command
“set command” on page 52 command
“hideSequenceNumbers parameter” on page 125 parameter
“maintainSequenceNumbers parameter” on page 140 parameter
“sequenceNumbers parameter” on page 182 parameter

sequenceNumberStyle parameter

The **sequenceNumberStyle** parameter may be used to query or set the style character that is used to display the sequence numbers in the current view.

Availability

query command
set command

Scope

The current view.

Syntax

```
query sequenceNumberStyle
set sequenceNumberStyle c
```

Parameters

c

Use the *c* parameter to specify the sequence number style character.

Description

The style attributes for the style character can be specified with the **styleAttributes** parameter.

Examples

```
query sequenceNumberStyle
set sequenceNumberStyle s
```

RELATED REFERENCES

“query command” on page 47 command
“resequence command” on page 50 command
“set command” on page 52 command
“hideSequenceNumbers parameter” on
page 125 parameter
“sequenceNumber parameter” on page 180
parameter
“sequenceNumbers parameter” parameter
“styleAttributes parameter” on page 187
parameter

sequenceNumbers parameter

The **sequenceNumbers** parameter may be used to query or set the starting column and width of the sequence numbers.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

sequenceNumbers is scoped to the current view’s document.
current.sequenceNumbers is scoped to the current view’s document.
default.sequenceNumbers is globally scoped.
install.sequenceNumbers is globally scoped.

Syntax

```
query sequenceNumbers
set sequenceNumbers [ default
                     | column width
                     ]

query current.sequenceNumbers
query default.sequenceNumbers
set default.sequenceNumbers [ install
                             | column width
                             ]

query install.sequenceNumbers
```

Parameters

default

If you specify the **default** parameter for the **set sequenceNumbers** command, then the current document will use the value of **default.sequenceNumbers** to determine the sequence numbers.

column width

If you specify the **width** and **column** parameters for the **set sequenceNumbers** command, then the specified width and column will be used to determine the current document's sequence numbers. If you specify the **width** and **column** parameters for the **set default.sequenceNumbers** command, then the specified width and column will be used to determine the sequence numbers for all of the documents that have **sequenceNumbers** set to **default**.

Use the *width* parameter to specify the width of the document's sequence numbers.

Use the *column* parameter to specify the starting column of the document's sequence numbers.

install

If you specify the **install** parameter for the **set default.sequenceNumbers** command, then all of the documents that have **sequenceNumbers** set to **default** will use the value of **install.sequenceNumbers** to determine the document's sequence numbers.

Description

If you do not specify any of the parameters for the **set sequenceNumbers** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.sequenceNumbers** command, then **install** is assumed.

The **query current.sequenceNumbers** command will return the sequence numbers column and width that are being used by the current document.

Examples

```
query sequenceNumbers
set sequenceNumbers 73 8
query current.sequenceNumbers
query default.sequenceNumbers
set default.sequenceNumbers install
query install.sequenceNumbers
```

RELATED REFERENCES

“query command” on page 47 command
“resequence command” on page 50 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“hideSequenceNumbers parameter” on page 125 parameter
“maintainSequenceNumbers parameter” on page 140 parameter
“sequenceNumber parameter” on page 180 parameter

show parameter

The **show** parameter can be used to determine if the current view's current element is a show element. A show element is a line of document text that is not saved with the file. The **show** parameter will return **on** if the current element is a show element for this view, **off** if the current element is not a show element and **other** if the current element is a show element for another view of the current view's document.

Availability

query command

Scope

The current view's current element.

Syntax

`query show`

Examples

`query show`

RELATED REFERENCES

"insertShow command" on page 40 command

"query command" on page 47 command

"save command" on page 50 command

"line parameter" on page 137 parameter

"lines parameter" on page 139 parameter

status parameter

The **status** parameter may be used to query or set the command status. Many commands set the status parameter after the command has completed to indicate the command completion status information. User defined commands may wish to set the status parameter to indicate to the command's completion status.

Availability

query command

set command

Scope

Global.

Syntax

`query status`

`set status value`

Parameters

value

Use the *value* parameter to set the value of the status parameter. There are no restrictions on *value*.

Examples

```
query status
set status save.failed
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command

statusLine parameter

The **statusLine** parameter may be used to query or set the visibility of the status line. The status line parameter is an information line that appears above the text window. It provides status information on the current row and column, the insertion mode, the current number of changes, and it indicates if the document view is readonly.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

statusLine is scoped to the current view.
current.statusLine is scoped to the current view.
default.statusLine is globally scoped.
install.statusLine is globally scoped.

Syntax

```
query statusLine
set statusLine [ default | on | off ]
query current.statusLine
query default.statusLine
set default.statusLine [ install | on | off ]
query install.statusLine
```

Parameters

default

on

If you specify the **default** parameter for the **set statusLine** command, then the current view will use the value of **default.statusLine** to determine the visibility of the status line. If you specify the **on** parameter for the **set statusLine** command, then the status line will be visible on the current view. If you specify the **on** parameter for the **set default.statusLine** command, then the status line will be visible for all views that have **statusLine** set to **default**.

off

If you specify the **off** parameter for the **set statusLine** command, then the status line will not be visible on the current view. If you specify the **off** parameter for the **set default.statusLine** command, then the status line will not be visible for all views that have **statusLine** set to **default**.

install

If you specify the **install** parameter for the **set default.statusLine** command, then all of the views that have **statusLine** set to **default** will use the value of **install.statusLine** to determine the visibility of the status line.

Examples

```
query statusLine
set statusLine off
query current.statusLine
query default.statusLine
set default.statusLine off
query install.statusLine
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

current parameter

“default parameter” on page 83 parameter

“install parameter” on page 129 parameter

style parameter

The **style** parameter may be used to query or set the style characters for the current view’s view of the current view’s element. The style attributes for the style characters are specified with the **styleAttributes** parameter. Style characters are normally set by the parser. The parser normally sets a string of style characters that is equal in length to the length of the element’s text. The editor will draw each character of element text with the style attributes of the style character that is in the corresponding position in the style string.

Availability

query command

set command

Scope

The current view’s view of the current view’s current element.

Syntax

```
query style
set style styleCharacters
```

Parameters

styleCharacters

Use the *styleCharacters* parameter to indicate the style characters that should be used to display the text on the current element..

Examples

```
query style
set style cccc__aaaa
```

RELATED REFERENCES

“query command” on page 47 command

“set command” on page 52 command

“styleAttributes parameter” parameter

“text parameter” on page 191 parameter

styleAttributes parameter

The **styleAttributes** parameter may be used to query or set the style attributes for a style character or for one of the built in styles.

Availability

query command

set command

Scope

The current view.

Syntax

```
query styleAttributes.{ styleCharacter
                        addedLines
                        background
                        default
                        deletedLines
                        emphasis
                        expandHide
                        lineNumbers
                        messageLine
                        prefixText
                        selection
                        statusLine
                        }
set styleAttributes.{ styleCharacter
                     addedLines
                     background
                     default
                     deletedLines
                     emphasis
                     expandHide
                     lineNumbers
                     messageLine
                     prefixText
                     selection
                     statusLine
                     }
[ foregroundColor
  foregroundGreen
  foregroundBlue
  backgroundRed
  backgroundGreen
  backgroundBlue
  [ underline ]
  [ outline ]
]
```

Parameters

<i>styleCharacter</i>	The <i>styleCharacter</i> qualifier is used to indicate the style character that you want to query or set. The style characters are used with the set style command. The '!' style character is used as the default style character. New text that has not yet been parsed is displayed with the default style character's attributes.
addedLines	Use the addedLines qualifier to indicate the style attributes that should be used to display added lines that are detected by the compare command.
background	Use the background qualifier to indicate that you want to query or set the background's style attributes. The background style is used to draw any part of the text area that does not display text elements.
default	Use the default qualifier to indicate the style attributes that should be used when no other style attributes are indicated. This may happen if the editor encounters a style character that is not defined. This style will also be used if no style characters are set for the element text.
deletedLines	Use the deletedLines qualifier to indicate the style attributes that should be used to display deleted lines that are detected by the compare command.
emphasis	Use the emphasis qualifier to indicate the style attributes that should be used to emphasize text. Text is normally emphasized by the findText command or locate command when a search is successful.
expandHide	Use the expandHide qualifier to indicate the style attributes that should be used to draw the expand/hide area. See the set expandHide command.
lineNumbers	Use the lineNumbers qualifier to indicate the style attributes that should be used to display the line numbers. See the set lineNumbers command.
messageLine	Use the messageLine qualifier to indicate the style attributes that should be used to display the message line.
prefixText	Use the prefixText qualifier to indicate the style attributes that should be used to display text that is typed into the prefix area.
selection	Use the selection qualifier to indicate the style attributes that should be used to display the text selection.
statusLine	Use the statusLine qualifier to indicate the style attributes that should be used to display the status line.
<i>foregroundRed</i>	Use the <i>foregroundRed</i> parameter to indicate the red component of the foreground color. <i>foregroundRed</i> must be an integer between 0 and 255.

<i>foregroundGreen</i>	Use the <i>foregroundGreen</i> parameter to indicate the green component of the foreground color. <i>foregroundGreen</i> must be an integer between 0 and 255.
<i>foregroundBlue</i>	Use the <i>foregroundBlue</i> parameter to indicate the blue component of the foreground color. <i>foregroundBlue</i> must be an integer between 0 and 255.
<i>backgroundRed</i>	Use the <i>backgroundRed</i> parameter to indicate the red component of the background color. <i>backgroundRed</i> must be an integer between 0 and 255.
<i>backgroundGreen</i>	Use the <i>backgroundGreen</i> parameter to indicate the green component of the background color. <i>backgroundGreen</i> must be an integer between 0 and 255.
<i>backgroundBlue</i>	Use the <i>backgroundBlue</i> parameter to indicate the blue component of the background color. <i>backgroundBlue</i> must be an integer between 0 and 255.
<i>underline</i>	Use the optional underline parameter to indicate that the text should be underlined.
<i>outline</i>	Use the optional outline parameter to indicate that the text should be outline.

Description

If you specify **set styleAttributes.styleCharacter** with no parameters, then the specified style character's style attributes will be cleared. If you specify one of the built in styles with no parameters, then the style attributes of the built in style will be restored to the default setting.

Examples

```
query styleAttributes.c
set styleAttributes.c 0 128 128 255 255 255
```

RELATED REFERENCES

“query command” on page 47 command
 “set command” on page 52 command
 “style parameter” on page 186 parameter
 “updateProfile.paletteAttributes parameter”
 on page 201 parameter

tabs parameter

The **tabs** parameter may be used to query or set the tab stops used by the **nextTabStop** and **prevTabStop** actions.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

tabs is scoped to the current view.

current.tabs is scoped to the current view.

default.tabs is globally scoped.

install.tabs is globally scoped.

Syntax

```
query tabs
set tabs [ default
          [ tabStop ] [ ... ] [ every tabIncrement ]
        ]
query current.tabs
query default.tabs
set default.tabs [ install
                  [ tabStop ] [ ... ] [ every tabIncrement ]
                ]
query install.tabs
```

Parameters

default

If you specify the **default** parameter for the **set tabs** command, then the current view will use the value of **default.tabs** to determine the tab stops that should be used in the current document view.

[*tabStop*] [...] [every *tabIncrement*]

If you specify the *tabStop* or the *tabIncrement* parameter for the **set tabs** command, then the specified tab stops will be used by the current view. If you specify the *tabStop* or the *tabIncrement* parameter for the **set default.tabs** command, then the specified tab stops will be used by all views that have **tabs** set to **default**. You can indicate specific tab stops with the *tabStop* parameter. Once the cursor is beyond the last *tabStop*, then additional tab stops will be determined by repeatedly adding *tabIncrement* to the last *tabStop*. An initial *tabStop* of 1 is always assumed even if it is not specified.

install

If you specify the **install** parameter for the **set default.tabs** command, then all of the views that have **tabs** set to **default** will use the value of **install.tabs** to determine the tab stops that should be used in the current document view.

Description

If you do not specify any of the parameters for the **set tabs** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.tabs** command, then **install** is assumed.

The **query current.tabs** command will return the tab stops that are being used by the current view.

Examples

```
query tabs
set tabs 1 4 12 every 8
query current.tabs
query default.tabs
set default.tabs 1 every 8
query install.tabs
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“updateProfile command” on page 55 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
nextTabStop action
prevTabStop action

text parameter

The **text** parameter may be used to query or set the current element’s text.

Availability

query command
set command

Scope

The current element of the current view.

Syntax

```
query text
set text elementText
```

Parameters

elementText

Use the *elementText* parameter to specify the text for the current element.

Examples

```
query text
set text This is some text for the current element.
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“length parameter” on page 137 parameter
“style parameter” on page 186 parameter

textAreaWidth parameter

The **textAreaWidth** parameter may be used to determine the width in pixels of the text area.

Availability

`query` command

Scope

The current view.

Syntax

```
query textAreaWidth
```

Examples

```
query textAreaWidth
```

RELATED REFERENCES

“`query` command” on page 47 command

“`pixelPosition` parameter” on page 155

parameter

“`position` parameter” on page 158 parameter

“`scroll` parameter” on page 179 parameter

“`textWidth` parameter” parameter

`textWidth` parameter

The **`textWidth`** parameter may be used to determine the width in pixels of the current element’s text.

Availability

`query` command

Scope

The current view’s current element.

Syntax

```
query textWidth
```

Examples

```
query textWidth
```

RELATED REFERENCES

“`query` command” on page 47 command

“`displayPosition` parameter” on page 85

parameter

“`pixelPosition` parameter” on page 155

parameter

“`position` parameter” on page 158 parameter

“`scroll` parameter” on page 179 parameter

“`textAreaWidth` parameter” on page 191

parameter

topExpanded parameter

The **topExpanded** parameter may be used to query or set the visibility of hidden elements between the top of the document and the first visible element. If you set **topExpanded** to **on**, then all of the hidden elements between the top of the document and the first visible element will be forced visible. If you set **topExpanded** to **off**, then all of the hidden elements between the top of the document and the first visible element will remain hidden.

Availability

query command

set command

Scope

The current view.

Syntax

```
query topExpanded
set topExpanded { on | off }
```

Parameters

on

Use the **on** parameter to indicate that the hidden elements between the top of the document and the first visible element should be forced visible..

off

Use the **off** parameter to indicate that the hidden elements between the top of the document and the first visible element remain hidden.

Description

The **topExpanded** attribute is set to **off** if the any of the following happens:

- the “**includedClasses parameter**” on **page 127** parameter is altered.
- the “**excludedClasses parameter**” on **page 89** parameter is altered.
- the “**markIncluded parameter**” on **page 147** parameter is altered.
- the “**markExcluded parameter**” on **page 143** paramter is altered.

Examples

```
query topExpanded
set topExpanded on
```

RELATED REFERENCES

“expandAll command” on page 33 command
“query command” on page 47 command
“set command” on page 52 command
“excludedClasses parameter” on page 89 parameter
“expanded parameter” on page 94 parameter
“includedClasses parameter” on page 127 parameter
“markIncluded parameter” on page 147 parameter
“markExcluded parameter” on page 143 parameter

updateProfile.baseProfile parameter

The **updateProfile.baseProfile** parameter may be used to query or set the base profile that will be used by the **updateProfile** command. The base profile governs the initial action assignments for key and mouse actions. It will also affect the prefix commands that are available with the **processPrefix** action.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.baseProfile is scoped to the current view.
current.updateProfile.baseProfile is scoped to the current view.
default.updateProfile.baseProfile is globally scoped.
install.updateProfile.baseProfile is globally scoped.

Syntax

```
query updateProfile.baseProfile
set updateProfile.baseProfile { default
                               | [ profile ]
                               }

query current.updateProfile.baseProfile
query default.updateProfile.baseProfile
set default.updateProfile.baseProfile { install
                                       | [ profile ]
                                       }

query install.updateProfile.baseProfile
```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.baseProfile** command, then the current view will use the value of **default.updateProfile.baseProfile** as the base profile for the **updateProfile** command.

profile

If you specify the *profile* parameter for the **set updateProfile.baseProfile** command, then for the current view, the **updateProfile** command will use the base profile indicated by *profile*. If you specify the *profile* parameter for the **set default.updateProfile.baseProfile** command, then for all document views that have the **updateProfile.baseProfile** parameter set to **default**, the **updateProfile** command will use the base profile indicated by *baseProfile*. The editor recognizes the following base profiles:

- lpex
- brief
- epm
- seu
- xedit
- ispf

install

If you specify the **install** parameter for the **set default.updateProfile.baseProfile** command, then all of the views that have **updateProfile.baseProfile** set to **default** will use the value of **install.updateProfile.baseProfile** to determine if the base profile that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.baseProfile** command will return the base profile that will be used, within the current view, by the **updateProfile** command. If you specify a base profile other than one of the ones understood by the editor, the editor will behave as though the **lpex** base profile was chosen.

Examples

```
query updateProfile.baseProfile
set updateProfile.baseProfile seu
query current.updateProfile.baseProfile
query default.updateProfile.baseProfile
set default.updateProfile.baseProfile lpex
query install.updateProfile.baseProfile
```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
“baseProfile parameter” on page 66
parameter
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“updateProfile.parser parameter” on page 204
parameter
“updateProfile.userActions parameter” on
page 211 parameter
“updateProfile.userCommands parameter” on
page 213 parameter
“updateProfile.userKeyActions parameter” on
page 215 parameter
“updateProfile.userMouseActions parameter”
on page 217 parameter
“updateProfile.userProfile parameter” on
page 219 parameter

updateProfile.extensions parameter

The **updateProfile.extensions** parameter can be used to list all of the file extensions that are associated with parsers. The **updateProfile.parserAssociation** parameter is used to indicate the parser that is associated with a specific file extension.

Availability

query command
current parameter
default parameter
install parameter

Scope

updateProfile.extensions is scoped to the current view.
current.updateProfile.extensions is scoped to the current view.
default.updateProfile.extensions is globally scoped.
install.updateProfile.extensions is globally scoped.

Syntax

```
query updateProfile.extensions  
query current.updateProfile.extensions  
query default.updateProfile.extensions  
query install.updateProfile.extensions
```

Description

The **query current.updateProfile.extensions** command will return the list of file extensions that are defined within the current view.

The **query updateProfile.extensions** command will return the list of file extensions that were specified with the **set updateProfile.parserAssociation** command.

The **query default.updateProfile.extensions** command will return the list of file extensions that were specified with the **set default.updateProfile.parserAssociation** command.

The **query install.updateprofile.extensions** command will return the list of installed file extensions. That is, the list of file extensions in the profile indicated by the **installProfile** parameter.

Examples

```
query updateProfile.extensions
query current.updateProfile.extensions
query default.updateProfile.extensions
query install.updateProfile.extensions
```

RELATED REFERENCES

“query command” on page 47 command

“updateProfile command” on page 55
command

“updateProfile.parserAssociation parameter”
on page 206 parameter

updateProfile.noParser parameter

The **updateProfile.noParser** parameter may be used to query or set if the **updateProfile** command should set a parser.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.noParser is scoped to the current view.

current.updateProfile.noParser is scoped to the current view.

default.updateProfile.noParser is globally scoped.

install.updateProfile.noParser is globally scoped.

Syntax

```
query updateProfile.noParser
set updateProfile.noParser [ default | on | off ]
query current.updateProfile.noParser
query default.updateProfile.noParser
set default.updateProfile.noParser [ install | on | off ]
query install.updateProfile.noParser
```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.noParser** command, then the current view will use the value of **default.updateProfile.noParser** to determine if the **updateProfile** command should set a parser.

on

If you specify the **on** parameter for the **set updateProfile.noParser** command, then the **updateProfile** command will not set a parser for the current view. If you specify the **on** parameter for the **set**

off

default.updateProfile.noParser command, then the **updateProfile** command will not set a parser for all of the document views that have **updateProfile.noParser** set to **default**.

If you specify the **off** parameter for the **set updateProfile.noParser** command, then the **updateProfile** command will attempt to set a parser for the current view. If you specify the **off** parameter for the **set default.updateProfile.noParser** command, then the **updateProfile** command attempt to set a parser for all of the document views that have **updateProfile.noParser** set to **default**.

install

If you specify the **install** parameter for the **set default.updateProfile.noParser** command, then all of the document views that have **updateProfile.noParser** set to **default** will use the value of **install.updateProfile.noParser** to determine if the **updateProfile** command should attempt to set a parser.

Description

If you do not specify any of the parameters for the **set updateProfile.noParser** command, then **default** is assumed.

If you do not specify any of the parameters for the **set default.updateProfile.noParser** command, then **install** is assumed.

The **query current.updateProfile.noParser** command will return **on** if, for the current view, the **updateProfile** command will attempt to set a parser and **off** if, for the current view, the **updateProfile** command will not set a parser.

Examples

```
query updateProfile.noParser
set updateProfile.noParser off
query current.updateProfile.noParser
query default.updateProfile.noParser
set default.updateProfile.noParser on
query install.updateProfile.noParser
```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“parser parameter” on page 155 parameter
“updateProfile.parser parameter” on page 204
parameter

updateProfile.palette parameter

The **updateProfile.palette** parameter may be used to query or set the color palette that will be used by the **updateProfile** command. The color palette is used to determine the colors that will be used by the editor.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.palette is scoped to the current view.
current.updateProfile.palette is scoped to the current view.
default.updateProfile.palette is globally scoped.
install.updateProfile.palette is globally scoped.

Syntax

```
query updateProfile.palette
set updateProfile.palette { default
                           | [ colorPalette ]
                           }

query current.updateProfile.palette
query default.updateProfile.palette
set default.updateProfile.palette { install
                                   | [ colorPalette ]
                                   }

query install.updateProfile.palette
```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.palette** command, then the current view will use the value of **default.updateProfile.palette** as the color palette for the **updateProfile** command.

colorPalette

If you specify the *colorPalette* parameter for the **set updateProfile.palette** command, then for the current view, the **updateProfile** command will use the color palette indicated by *colorPalette*. If you specify the *colorPalette* parameter for the **set default.updateProfile.palette** command, then for all document views that have the **updateProfile.palette** parameter set to **default**, the **updateProfile** command will use the color palette indicated by *colorPalette*. The available palettes can be determined by querying the **current.updateProfile.palettes** parameter.

install

If you specify the **install** parameter for the **set default.updateProfile.palette** command, then all of the views that have **updateProfile.palette** set to **default** will use the value of **install.updateProfile.palette** to determine if the color palette that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.palette** command will return the palette that will be used, within the current view, by the **updateProfile** command.

If you specify a palette other than one of the ones understood by the editor, the editor will use its default color settings.

Examples

```
query updateProfile.palette
set updateProfile.palette black
query current.updateProfile.palette
query default.updateProfile.palette
set default.updateProfile.palette white
query install.updateProfile.palette
```

RELATED REFERENCES

“updateProfile command” on page 55
command

“query command” on page 47 command

“set command” on page 52 command

current parameter

“default parameter” on page 83 parameter

“install parameter” on page 129 parameter

“palette parameter” on page 154 parameter

“styleAttributes parameter” on page 187

parameter

“updateProfile.paletteAttributes parameter”
on page 201 parameter

“updateProfile.palettes parameter” on
page 203 parameter

updateProfile.paletteAttributes parameter

The **updateProfile.paletteAttributes** parameter may be used to query or set the style attributes that are to be used for a specified style and palette when the **updateProfile** command is issued.

Availability

query command

set command

current parameter

default parameter

install parameter

Scope

updateProfile.paletteAttributes.style.palette is scoped to the current view.

current.updateProfile.paletteAttributes.style.palette is scoped to the current view.

default.updateProfile.paletteAttributes.style.palette is globally scoped.

install.updateProfile.paletteAttributes.style.palette is globally scoped.

Syntax

```
query updateProfile.paletteAttributes.style.palette
set updateProfile.paletteAttributes.style.palette { default
                                                    | [ styleAttributes ]
                                                    }

query current.updateProfile.paletteAttributes.style.palette
query default.updateProfile.paletteAttributes.style.palette
set default.updateProfile.paletteAttributes.style.palette { install
                                                            | [ styleAttributes ]
                                                            }

query install.updateProfile.paletteAttributes.style.palette
```

Parameters

style

Use the *style* parameter to indicate the style. The following styles are available:

- addedLines
- background
- default
- deletedLines
- emphasis
- expandHide
- lineNumbers
- messageLine
- prefixText
- selection
- statusLine

Refer to the **styleAttributes** parameter for more information on these styles.

<i>palette</i>	Use the <i>palette</i> parameter to indicate the palette for which you wish to define style attributes. When defining a palette you must at a minimum indicate the style attributes for the default style. It is those palettes for which the default style is defined that are returned by the current.updateProfile.palettes parameter.
default	If you specify the default parameter for the set updateProfile.paletteAttributes.style.palette command, then the current view will use the value of default.updateProfile.paletteAttributes.style.palette to determine the style attributes when the updateProfile command is issued.
<i>styleAttributes</i>	If you specify the <i>styleAttributes</i> parameter for the set updateProfile.paletteAttributes.style.palette command, then for the current view, the updateProfile command will use <i>styleAttributes</i> as the style attributes for the style and palette indicated by <i>style</i> and <i>palette</i> . If you specify the <i>styleAttributes</i> parameter for the set default.updateProfile.paletteAttributes.style.palette command, then for all document views that have the updateProfile.paletteAttributes.style.palette parameter set to default , the updateProfile command will use <i>styleAttributes</i> as the style attributes for the style and palette indicated by <i>style</i> and <i>palette</i> . For information on the valid syntax of <i>styleAttributes</i> , refer to the styleAttributes parameter.
install	If you specify the install parameter for the set default.updateProfile.paletteAttributes.style.palette command, then all of the views that have updateProfile.paletteAttributes.style.palette set to default will use the value of install.updateProfile.paletteAttributes.style.palette to determine the style attributes when the updateProfile command is issued.

Description

The query **current.updateProfile.paletteAttributes.style.palette** command will return the style attributes that the **updateProfile** command will use for the current view for the specified style and palette.

If you do not specify any parameters for the **set updateProfile.paletteAttributes.style.palette** command, then no style attributes will be used for the specified style and palette.

If you do not specify any parameters for the **set default.updateProfile.paletteAttributes.style.palette** command, then no default style attributes will be used for the specified style and palette.

Examples

```
query updateProfile.paletteAttributes.default.white
set updateProfile.paletteAttributes.default.white 0 0 170 255 255 255
set updateProfile.paletteAttributes.default.white default
set updateProfile.paletteAttributes.lineNumbers.black
query current.updateProfile.paletteAttributes.default.gray
query default.updateProfile.paletteAttributes.background.white
set default.updateProfile.paletteAttributes.background.white 0 0 0 204 204 204
set default.updateProfile.paletteAttributes.background.white install
set default.updateProfile.paletteAttributes.background.white
query install.updateProfile.paletteAttributes.default.white
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“updateProfile command” on page 55 command
“palette parameter” on page 154 parameter
“styleAttributes parameter” on page 187 parameter
“updateProfile.palette parameter” on page 199 parameter
“updateProfile.palettes parameter” parameter

updateProfile.palettes parameter

The **updateProfile.palettes** parameter can be used to list all of the color palettes that are available to the **updateProfile** command.

Availability

query command
current parameter
default parameter
install parameter

Scope

updateProfile.palettes is scoped to the current view.
current.updateProfile.palettes is scoped to the current view.
default.updateProfile.palettes is globally scoped.
install.updateProfile.palettes is globally scoped.

Syntax

```
query updateProfile.palettes
query current.updateProfile.palettes
query default.updateProfile.palettes
query install.updateProfile.palettes
```

Description

The **query current.updateProfile.palettes** command will return the palettes that are available, within the current view, to the **updateProfile** command.

The **query updateProfile.palettes** command will return the list of palettes that were specified with the **set updateProfile.paletteAttributes.default** command.

The **query default.updateProfile.palettes** command will return the list of color palettes that were specified with the **set default.updateProfile.paletteAttributes.default** command.

The **query install.updateProfile.palettes** command will return the list of installed palettes. That is, the list of palettes in the profile indicated by the **installProfile** parameter.

Examples

```
query updateProfile.palettes
query current.updateProfile.palettes
query default.updateProfile.palettes
query install.updateProfile.palettes
```

RELATED REFERENCES

“query command” on page 47 command
“updateProfile command” on page 55
command
“updateProfile.palette parameter” on
page 199 parameter
“updateProfile.paletteAttributes parameter”
on page 201 parameter

updateProfile.parser parameter

The **updateProfile.parser** parameter may be used to query or set the parser that will be used by the **updateProfile** command. The parser should be one of the parsers returned by the **current.updateProfile.parsers** parameter.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.parser is scoped to the current view.
current.updateProfile.parser is scoped to the current view.
default.updateProfile.parser is globally scoped.
install.updateProfile.parser is globally scoped.

Syntax

```
query updateProfile.parser
set updateProfile.parser { associated
                        default
                        [ parser ]
                        }

query current.updateProfile.parser
query default.updateProfile.parser
set default.updateProfile.parser { associated
                                install
                                [ parser ]
                                }

query install.updateProfile.parser
```

Parameters

associated

If you specify the **associated** parameter for the **set updateProfile.parser** command, then the **updateProfile** command will choose, for the current view, the parser based that is associated with the document's file name extension. If you specify the **associated** parameter for the **set default.updateProfile.parser** command, then for all document views that have **updateProfile.parser** set to **default**, the **updateProfile** command will choose the parser that is associated with the document's file name extension. Parser associations are specified with the **updateProfile.parserAssociation** parameter.

default

If you specify the **default** parameter for the **set updateProfile.parser** command, then the current view will use the value of **default.updateProfile.parser** as the parser for the **updateProfile** command.

parser

If you specify the *parser* parameter for the **set updateProfile.parser** command, then within the current view, the **updateProfile** command will use the parser indicated by *parser*. If you specify the *parser* parameter for the **set default.updateProfile.parser** command, then within all document views that have the **updateProfile.parser** set to **default**, the **updateProfile** command will use the parser indicated by *parser*. The parser you specify should be one of the parsers returned by the **updateProfile.parsers** parameter.

install

If you specify the **install** parameter for the **set default.updateProfile.parser** command, then all of the views that have **updateProfile.parser** set to **default** will use the value of **install.updateProfile.parser** to determine if the parser that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.parser** command will return the parser that will be used, within the current view, by the **updateProfile** command.

Examples

```
query updateProfile.parser
set updateProfile.parser java
query current.updateProfile.parser
query default.updateProfile.parser
set default.updateProfile.parser associated
query install.updateProfile.parser
```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“parser parameter” on page 155 parameter
“updateProfile.baseProfile parameter” on
page 194 parameter
“updateProfile.noParser parameter” on
page 197 parameter
“updateProfile.parserAssociation parameter”
parameter
“updateProfile.parsers parameter” on
page 210 parameter
“updateProfile.userActions parameter” on
page 211 parameter
“updateProfile.userCommands parameter” on
page 213 parameter
“updateProfile.userKeyActions parameter” on
page 215 parameter
“updateProfile.userMouseActions parameter”
on page 217 parameter
“updateProfile.userProfile parameter” on
page 219 parameter

updateProfile.parserAssociation parameter

The **updateProfile.parserAssociation** parameter may be used to query or set the parser that is associated with a file extension. When the **updateProfile** command is issued on a document view that has **current.updateProfile.parser** set to **associated**, the parser that is associated with the files file extension is used to parse the document view.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.parserAssociation.fileExtension is scoped to the current view.
current.updateProfile.parserAssociation.fileExtension is scoped to the current view.
default.updateProfile.parserAssociation.fileExtension is globally scoped.
install.updateProfile.parserAssociation.fileExtension is globally scoped.

Syntax

```
query updateProfile.parserAssociation.fileExtension
set updateProfile.parserAssociation.fileExtension { default
                                                    | [ parser ]
                                                    }
query current.updateProfile.parserAssociation.fileExtension
```

```

query default.updateProfile.parserAssociation.fileExtension
set default.updateProfile.parserAssociation.fileExtension { install
                                                    | [ parser ]
                                                    }
query install.updateProfile.parserAssociation.fileExtension

```

Parameters

fileExtension

Use the *fileExtension* parameter to indicate the file extension.

default

If you specify the **default** parameter for the **set**

updateProfile.parserAssociation.fileExtension command, then the current view will use the value of **default.updateProfile.parserAssociation.fileExtension** to determine the parser associated with *fileExtension* when the **updateProfile** command is issued.

parser

If you specify the *parser* parameter for the **set** **updateProfile.parserAssociation.fileExtension** command, then for the current view, the **updateProfile** command will use *parser* as the parser associated with the file extension indicated by *fileExtension*. If you specify the *parser* parameter for the **set**

default.updateProfile.parserAssociation.fileExtension command, then for all document views that have the

updateProfile.parserAssociation.fileExtension parameter set to **default**, the **updateProfile** command will use *parser* as the parser associated with the file extension indicated by *fileExtension*. The *parser* parameter should be one of the parsers returned by the **current.updateProfile.parsers** parameter.

install

If you specify the **install** parameter for the **set**

default.updateProfile.parserAssociation.fileExtension command, then all of the views that have **updateProfile.parserAssociation.fileExtension** set to **default** will use the value of **install.updateProfile.parserAssociation.fileExtension** to determine the parser associated with *fileExtension* when the **updateProfile** command is issued.

Description

The **query current.updateProfile.parserAssociation.fileExtension** command will return the parser that the **updateProfile** command will associate, for the current view, with the file extension specified by *fileExtension*.

If you do not specify any parameters for the **set** **updateProfile.parserAssociation.fileExtension** command, then no parser will be associated with the specified file extension.

If you do not specify any parameters for the **set** **default.updateProfile.parserAssociation.fileExtension** command, then no default parser will be associated with the specified file extension.

Examples

```
query updateProfile.parserAssociation.java
set updateProfile.parserAssociation.java java
set updateProfile.parserAssociation.java default
set updateProfile.parserAssociation.java
query current.updateProfile.parserAssociation.java
query default.updateProfile.parserAssociation.java
set default.updateProfile.parserAssociation.java java
set default.updateProfile.parserAssociation.java install
set default.updateProfile.parserAssociation.java
query install.updateProfile.parserAssociation.java
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“updateProfile command” on page 55 command
“parser parameter” on page 155 parameter
“updateProfile.extensions parameter” on page 196 parameter
“updateProfile.noParser parameter” on page 197 parameter
“updateProfile.parserClass parameter” parameter
“updateProfile.parsers parameter” on page 210 parameter
“updateProfile.parser parameter” on page 204 parameter

updateProfile.parserClass parameter

The **updateProfile.parserClass** parameter may be used to query or set the class name for a parser. A parser class name must be a java class that implements the **com.ibm.lpex.core.LpexParser** interface.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.parserClass.parser is scoped to the current view.
current.updateProfile.parserClass.parser is scoped to the current view.
default.updateProfile.parserClass.parser is globally scoped.
install.updateProfile.parserClass.parser is globally scoped.

Syntax

```
query updateProfile.parserClass.parser
set updateProfile.parserClass.parser { default
                                     | [ className ]
                                     }

query current.updateProfile.parserClass.parser
query default.updateProfile.parserClass.parser
set default.updateProfile.parserClass.parser { install
```

```

| [ className ]
}
query install.updateProfile.parserClass.parser

```

Parameters

parser
default

Use the *parser* parameter to the parser.
If you specify the **default** parameter for the **set updateProfile.parserClass.parser** command, then the current view will use the value of **default.updateProfile.parserClass.parser** to determine the class name for the parser indicated by *parser* when the **updateProfile** command is issued.

className

If you specify the *className* parameter for the **set updateProfile.parserClass.parser** command, then for the current view, the **updateProfile** command will use *className* as the class name for the parser indicated by *parser*. If you specify the *className* parameter for the **set default.updateProfile.parserClass.parser** command, then for all document views that have the **updateProfile.parserClass.parser** parameter set to **default**, the **updateProfile** command will use *className* as the class name for the parser indicated by *parser*. The *className* parameter should be the name of a java class that implements the **com.ibm.lpex.core.LpexParser** interface.

install

If you specify the **install** parameter for the **set default.updateProfile.parserClass.parser** command, then all of the views that have **updateProfile.parserClass.parser** set to **default** will use the value of **install.updateProfile.parserClass.parser** to determine the class name for the parser indicated by *parser* when the **updateProfile** command is issued.

Description

The **query current.updateProfile.parserClass.parser** command will return the class name that the **updateProfile** command will use, for the current view, for the parser specified by *parser*.

If you do not specify any parameters for the **set updateProfile.parserClass.parser** command, then the specified parser will not be available, for the current view, to the **updateProfile** command.

If you do not specify any parameters for the **set default.updateProfile.parserClass.parser** command, then the specified parser will not be available, for all document views that have **updateProfile.parserClass.parser** set to **default**, to the **updateProfile** command.

Examples

```

query updateProfile.parserClass.java
set updateProfile.parserClass.java com.ibm.lpex.java.JavaParser
set updateProfile.parserClass.java default

```

```
set updateProfile.parserClass.java
query current.updateProfile.parserClass.java
query default.updateProfile.parserClass.java
set default.updateProfile.parserClass.java com.ibm.lplex.java.JavaParser
set default.updateProfile.parserClass.java install
set default.updateProfile.parserClass.java
query install.updateProfile.parserClass.java
```

RELATED REFERENCES

“query command” on page 47 command
“set command” on page 52 command
“updateProfile command” on page 55 command
“parser parameter” on page 155 parameter
“updateProfile.extensions parameter” on page 196 parameter
“updateProfile.noParser parameter” on page 197 parameter
“updateProfile.parserAssociation parameter” on page 206 parameter
“updateProfile.parsers parameter” parameter
“updateProfile.parser parameter” on page 204 parameter

updateProfile.parsers parameter

The **updateProfile.parsers** parameter can be used to list all of the parsers that are available to the updateProfile command.

Availability

query command
current parameter
default parameter
install parameter

Scope

updateProfile.parsers is scoped to the current view.
current.updateProfile.parsers is scoped to the current view.
default.updateProfile.parsers is globally scoped.
install.updateProfile.parsers is globally scoped.

Syntax

```
query updateProfile.parsers
query current.updateProfile.parsers
query default.updateProfile.parsers
query install.updateProfile.parsers
```

Description

The **query current.updateProfile.parsers** command will return the parsers that will be available, within the current view, by the **updateProfile** command.

The **query updateProfile.parsers** command will return the list of parsers that were specified with the **set updateProfile.parserClass** command.

The **query default.updateProfile.parsers** command will return the list of parsers that were specified with the **set default.updateProfile.parserClass** command.

The **query install.updateprofile.parsers** command will return the list of installed parsers. That is, the list of parsers in the profile indicated by the **installProfile** parameter.

Examples

```
query updateProfile.parsers
query current.updateProfile.parsers
query default.updateProfile.parsers
query install.updateProfile.parsers
```

RELATED REFERENCES

“query command” on page 47 command

“updateProfile command” on page 55 command

“updateProfile.parserClass parameter” on page 208 parameter

updateProfile.userActions parameter

The **updateProfile.userActions** parameter may be used to query or set the user actions that will be used by the **updateProfile** command. The user actions are a list of action/class name pairs that you wish to define when the **updateProfile** command is issued.

Availability

query command

set command

current parameter

default parameter

install parameter

Scope

updateProfile.userActions is scoped to the current view.

current.updateProfile.userActions is scoped to the current view.

default.updateProfile.userActions is globally scoped.

install.updateProfile.userActions is globally scoped.

Syntax

```
query updateProfile.userActions
set updateProfile.userActions { default
                                | [ action className ] [...]
                                }

query current.updateProfile.userActions
query default.updateProfile.userActions
set default.updateProfile.userActions { install
                                        | [ action className ] [...]
                                        }

query install.updateProfile.userActions
```

Parameters

default

[*action className*] [...]

install

If you specify the **default** parameter for the **set updateProfile.userActions** command, then the current view will use the value of **default.updateProfile.userActions** as the user actions for the **updateProfile** command. If you specify a list of action/class name pairs for the **set updateProfile.userActions** command, then within the current view, the **updateProfile** command will use that list of user actions. If you specify the *userActions* parameter for the **set default.updateProfile.userActions** command, then within all document views that have the **updateProfile.userActions** set to **default**, the **updateProfile** command will use that list of user actions. The action parameter should be the name of the action that you want to define. The *className* parameter should be the name of a java class that implements the **com.ibm.lpex.core.LpexAction** interface. If you specify the **install** parameter for the **set default.updateProfile.userActions** command, then all of the views that have **updateProfile.userActions** set to **default** will use the value of **install.updateProfile.userActions** to determine if the user actions that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.userActions** command will return the action/class name pairs that will be used, within the current view, by the **updateProfile** command.

Examples

```
query updateProfile.userActions
set updateProfile.userActions java
query current.updateProfile.userActions test com.ibm.lpex.samples.TestAction
query default.updateProfile.userActions
set default.updateProfile.userActions install
query install.updateProfile.userActions
```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
“actionClass parameter” on page 63
parameter
“actions parameter” on page 65 parameter
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“updateProfile.baseProfile parameter” on
page 194 parameter
“updateProfile.parser parameter” on page 204
parameter
“updateProfile.userCommands parameter”
parameter
“updateProfile.userKeyActions parameter” on
page 215 parameter
“updateProfile.userMouseActions parameter”
on page 217 parameter
“updateProfile.userProfile parameter” on
page 219 parameter

updateProfile.userCommands parameter

The **updateProfile.userCommands** parameter may be used to query or set the user commands that will be used by the **updateProfile** command. The user commands are a list of command/class name pairs that you wish to define when the **updateProfile** command is issued.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.userCommands is scoped to the current view.
current.updateProfile.userCommands is scoped to the current view.
default.updateProfile.userCommands is globally scoped.
install.updateProfile.userCommands is globally scoped.

Syntax

```
query updateProfile.userCommands
set updateProfile.userCommands { default
                                | [ command className ] [...]
                                }

query current.updateProfile.userCommands
query default.updateProfile.userCommands
set default.updateProfile.userCommands { install
                                         | [ command className ] [...]
                                         }

query install.updateProfile.userCommands
```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.userCommands** command, then the current view will use the value of **default.updateProfile.userCommands** as the user commands for the **updateProfile** command.

[*command className*] [...]

If you specify a list of command/class name pairs for the **set updateProfile.userCommands** command, then within the current view, the **updateProfile** command will use that list of user commands. If you specify a list of command/class name pairs for the **set default.updateProfile.userCommands** command, then within all document views that have the **updateProfile.userCommands** set to **default**, the **updateProfile** command will use that list of user commands. The *command* parameter is the name of the command that you want to define. The *className* parameter should be the name of a java class that implements the **com.ibm.lpex.core.LpexCommand** interface.

install

If you specify the **install** parameter for the **set default.updateProfile.userCommands** command, then all of the views that have **updateProfile.userCommands** set to **default** will use the value of **install.updateProfile.userCommands** to determine if the user commands that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.userCommands** command will return the list of command/class name pairs that will be used, within the current view, by the **updateProfile** command.

Examples

```
query updateProfile.userCommands
set updateProfile.userCommands test com.ibm.lpex.samples.TestCommand
query current.updateProfile.userCommands
query default.updateProfile.userCommands
set default.updateProfile.userCommands install
query install.updateProfile.userCommands
```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
“commandClass parameter” on page 71
parameter
“commands parameter” on page 74
parameter
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“updateProfile.baseProfile parameter” on
page 194 parameter
“updateProfile.parser parameter” on page 204
parameter
“updateProfile.userActions parameter” on
page 211 parameter
“updateProfile.userKeyActions parameter”
parameter
“updateProfile.userMouseActions parameter”
on page 217 parameter
“updateProfile.userProfile parameter” on
page 219 parameter

updateProfile.userKeyActions parameter

The **updateProfile.userKeyActions** parameter may be used to query or set the user key actions that will be used by the **updateProfile** command. The user key actions are a list of key/action pairs that you wish to define when the **updateProfile** command is issued.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.userKeyActions is scoped to the current view.
current.updateProfile.userKeyActions is scoped to the current view.
default.updateProfile.userKeyActions is globally scoped.
install.updateProfile.userKeyActions is globally scoped.

Syntax

```
query updateProfile.userKeyActions
set updateProfile.userKeyActions { default
                                   | [ key action ] [...]
                                   }

query current.updateProfile.userKeyActions
query default.updateProfile.userKeyActions
set default.updateProfile.userKeyActions { install
```

```

        | [ key action ] [...]
      }
query install.updateProfile.userKeyActions

```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.userKeyActions** command, then the current view will use the value of **default.updateProfile.userKeyActions** as the user commands for the **updateProfile** command.

[key action] [...]

If you specify a list of key/action pairs for the **set updateProfile.userKeyActions** command, then within the current view, the **updateProfile** command will use that list of user key actions. If you specify a list of key/action pairs for the **set default.updateProfile.userKeyActions** command, then within all document views that have the **updateProfile.userKeyActions** set to **default**, the **updateProfile** command will use that list of user key actions. The *key* parameter is the key that you want to define. Refer to the **keyAction** parameter to see the valid syntax for defining a key. The *action* parameter should be a valid action name.

install

If you specify the **install** parameter for the **set default.updateProfile.userKeyActions** command, then all of the views that have **updateProfile.userKeyActions** set to **default** will use the value of **install.updateProfile.userKeyActions** to determine if the user key actions that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.userKeyActions** command will return the list of key/action pairs that will be used, within the current view, by the **updateProfile** command.

Examples

```

query updateProfile.userKeyActions
set updateProfile.userKeyActions a-backSpace.t.p.secondary undo
query current.updateProfile.userKeyActions
query default.updateProfile.userKeyActions
set default.updateProfile.userKeyActions install
query install.updateProfile.userKeyActions

```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“keyAction parameter” on page 133
parameter
“keys parameter” on page 136 parameter
“updateProfile.baseProfile parameter” on
page 194 parameter
“updateProfile.parser parameter” on page 204
parameter
“updateProfile.userActions parameter” on
page 211 parameter
“updateProfile.userCommands parameter” on
page 213 parameter
“updateProfile.userMouseActions parameter”
parameter
“updateProfile.userProfile parameter” on
page 219 parameter

updateProfile.userMouseActions parameter

The **updateProfile.userMouseActions** parameter may be used to query or set the user mouse actions that will be used by the **updateProfile** command. The user mouse actions are a list of mouse event/action pairs that you wish to define when the **updateProfile** command is issued.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.userMouseActions is scoped to the current view.
current.updateProfile.userMouseActions is scoped to the current view.
default.updateProfile.userMouseActions is globally scoped.
install.updateProfile.userMouseActions is globally scoped.

Syntax

```
query updateProfile.userMouseActions
set updateProfile.userMouseActions { default
                                     | [ mouseEvent action ] [...]
                                     }

query current.updateProfile.userMouseActions
query default.updateProfile.userMouseActions
set default.updateProfile.userMouseActions { install
                                             | [ mouseEvent action ] [...]
                                             }

query install.updateProfile.userMouseActions
```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.userMouseActions** command, then the current view will use the value of

default.updateProfile.userMouseActions as the user commands for the **updateProfile** command.

[*mouseEvent action*] [...]

If you specify a list of mouse event/action pairs for the **set**

updateProfile.userMouseActions command, then within the current view, the

updateProfile command will use that list of user mouse actions. If you specify a list of mouse event/action pairs for the **set**

default.updateProfile.userMouseActions command, then within all document views that have the

updateProfile.userMouseActions set to **default**, the **updateProfile** command will use that list of user mouse actions. The

mouseEvent parameter is the mouse event that you want to define. Refer to the

mouseAction parameter to see the valid syntax for defining a mouse event. The *action* parameter should be a valid action name.

install

If you specify the **install** parameter for the **set default.updateProfile.userMouseActions** command, then all of the views that have **updateProfile.userMouseActions** set to **default** will use the value of **install.updateProfile.userMouseActions** to determine if the user mouse actions that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.userMouseActions** command will return the list of mouse events/action pairs that will be used, within the current view, by the **updateProfile** command.

Examples

```
query updateProfile.userMouseActions
set updateProfile.userMouseActions 1-pressed.1.t.p cursorToMouse
query current.updateProfile.userMouseActions
query default.updateProfile.userMouseActions
set default.updateProfile.userMouseActions install
query install.updateProfile.userMouseActions
```


RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“mouseAction parameter” on page 151
parameter
“mouseEvents parameter” on page 153
parameter
“updateProfile.baseProfile parameter” on
page 194 parameter
“updateProfile.parser parameter” on page 204
parameter
“updateProfile.userActions parameter” on
page 211 parameter
“updateProfile.userCommands parameter” on
page 213 parameter
“updateProfile.userKeyActions parameter” on
page 215 parameter
“updateProfile.userProfile parameter”
parameter

updateProfile.userProfile parameter

The **updateProfile.userProfile** parameter may be used to query or set the user profile that will be used by the **updateProfile** command. The user profile should be the name of a java class that contains the following method:

```
public static void userProfile(LpexView lpexView)
```

When the updateProfile command is issued, this method will be called.

Availability

query command
set command
current parameter
default parameter
install parameter

Scope

updateProfile.userProfile is scoped to the current view.
current.updateProfile.userProfile is scoped to the current view.
default.updateProfile.userProfile is globally scoped.
install.updateProfile.userProfile is globally scoped.

Syntax

```
query updateProfile.userProfile
set updateProfile.userProfile { default
                               | [ className ]
                               }

query current.updateProfile.userProfile
query default.updateProfile.userProfile
```

```

set default.updateProfile.userProfile { install
                                     | [ className ]
                                     }
query install.updateProfile.userProfile

```

Parameters

default

If you specify the **default** parameter for the **set updateProfile.userProfile** command, then the current view will use the value of **default.updateProfile.userProfile** as the user profile for the **updateProfile** command.

className

If you specify the *className* parameter for the **set updateProfile.userProfile** command, then within the current view, the **updateProfile** command will use the class indicated by *className* as the user profile.. If you specify the *className* parameter for the **set default.updateProfile.userProfile** command, then within all document views that have the **updateProfile.userProfile** set to **default**, the **updateProfile** command will use the class indicated by *className* as the user profile.

install

If you specify the **install** parameter for the **set default.updateProfile.userProfile** command, then all of the views that have **updateProfile.userProfile** set to **default** will use the value of **install.updateProfile.userProfile** to determine if the user profile that should be used by the **updateProfile** command.

Description

The **query current.updateProfile.userProfile** command will return the class name of the user profile that will be used, within the current view, by the **updateProfile** command.

Examples

```

query updateProfile.userProfile
set updateProfile.userProfile com.ibm.lpex.samples.TestUserProfile
query current.updateProfile.userProfile
query default.updateProfile.userProfile
set default.updateProfile.userProfile install
query install.updateProfile.userProfile

```

RELATED REFERENCES

“updateProfile command” on page 55
command
“query command” on page 47 command
“set command” on page 52 command
current parameter
“default parameter” on page 83 parameter
“install parameter” on page 129 parameter
“updateProfile.baseProfile parameter” on
page 194 parameter
“updateProfile.parser parameter” on page 204
parameter
“updateProfile.userActions parameter” on
page 211 parameter
“updateProfile.userCommands parameter” on
page 213 parameter
“updateProfile.userKeyActions parameter” on
page 215 parameter
“updateProfile.userMouseActions parameter”
on page 217 parameter

visible parameter

The **visible** parameter can be used to determine if the current view’s current element is visible in the current view. The **visible** parameter will return **on** if the current element is visible and **off** if the current element is not visible. The following parameters can affect the visibility of an element:

- elementClasses
- excludedClasses
- expanded
- forceAllVisible
- forceVisible
- markExcluded
- markIncluded
- show
- topExpanded

Availability

query command

Scope

The current view’s current element.

Syntax

query visible

Examples

query visible

RELATED REFERENCES

“query command” on page 47 command
“elementClasses parameter” on page 87
parameter
“excludedClasses parameter” on page 89
parameter
“expanded parameter” on page 94 parameter
“forceAllVisible parameter” on page 120
parameter
“forceVisible parameter” on page 120
parameter
“markExcluded parameter” on page 143
parameter
“markIncluded parameter” on page 147
parameter
“show parameter” on page 184 parameter
“topExpanded parameter” on page 193
parameter

Default editor actions

backSpace	<p>The backSpace action will do one of the following:</p> <ul style="list-style-type: none">• If the cursor is at the top of the file or there is not a previous line that is visible or not a show line, then nothing is done.• If the view is readonly, then nothing is done.• If there is a stream selection in the current view, then the selected text is deleted.• If the cursor is not at the start of a line, then the cursor is moved one character to the left, and the character at the new cursor position is deleted.• If the cursor is at the start of a line then the cursor is moved to the end of the previous visible line that is not a show line. The new current line is then joined with the old current line.
blockCopy	<p>If the view is not readonly and there is some visible text selected, then the blockCopy action will copy the selected text to the current cursor position. Otherwise, the blockCopy action will do nothing. After the text has been copied, the copied text will be reselected.</p>
blockDelete	<p>If the view is not readonly and the view contains some visible selected text, then the blockDelete action will delete the selected text.</p>
blockFill	<p>If the view is not readonly and the view contains some visible selected text, then the blockFill action will ask the user to specify a character and then it will fill the selected area with the specified character.</p>
blockLowerCase	<p>If the view is not readonly and the view contains some visible selected text, then the blockLowerCase action will change the selected text to lower case.</p>

blockMarkBottom	<p>If there is no selected text within the current view, then the blockMarkBottom action will select text from the current position to the end of the file using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkBottom action will extend the block selection to the end of the file.</p>
blockMarkCharacter	<p>If there is a character selection within the current view, then the blockMarkCharacter action will extend the character selection to the current cursor position. If there is no character selection within the current view, then the blockMarkCharacter action will select the character at the current cursor position with the character block type.</p>
blockMarkDown	<p>If there is no selected text within the current view, then the blockMarkDown action will select text from the current cursor position to the corresponding position on the next visible line using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkDown action will extend the block selection to the next visible line.</p>
blockMarkElement	<p>If there is an element selection within the current view, then the blockMarkElement action will extend the element selection to the current line. If there is no element selection within the current view, then the blockMarkElement action will select the current line with the element block type.</p>
blockMarkElementAtMouse	<p>The blockMarkElementAtMouse action will move the cursor to the mouse position and select the element at the new cursor position. If there is already a block selection, then the block selection is removed.</p>
blockMarkEnd	<p>If there is no selected text within the current view, then the blockMarkEnd action will select text from the current position to the end of the current line using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkEnd action will extend the block selection to the end of the current line.</p>
blockMarkHome	<p>If there is no selected text within the current view, then the blockMarkHome action will select text from the current position to the start of the current line using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkHome action will extend the block selection to the start of the current line.</p>
blockMarkLeft	<p>First if the current selection is an element selection, then it is cleared. If there is no selected text in the current view, then the current.block.defaultType setting is used to select the current character. If the current.block.defaultType setting is element, then stream is used. The current selection is then extended one character to the left. If the current selection type is character or stream and the cursor is at the start of the current line, then the selection is extended to include the last character on the previous visible line. If the current selection type is rectangle and the cursor is at the start of the line, then the selection is left unchanged.</p>

blockMarkNextWord	First if the current selection is an element selection, then it is cleared. If there is no selected text in the current view, then the current.block.defaultType setting is used to select the current character. If the current.block.defaultType setting is element, then stream is used. The current selection is then extended to include the next word.
blockMarkPageDown	If there is no selected text in the current view, then the current.block.defaultType setting is used to select a page of text starting at the current cursor position. If there is selected text in the current view, then the current selection is extended one page down from the current cursor position.
blockMarkPageLeft	First if the current selection is an element selection, then it is cleared. If there is no selected text in the current view, then the current.block.defaultType setting is used to select the current character. If the current.block.defaultType setting is element, then stream is used. The current selection is then extended one page to the left.
blockMarkPageRight	First if the current selection is an element selection, then it is cleared. If there is no selected text in the current view, then the current.block.defaultType setting is used to select the current character. If the current.block.defaultType setting is element, then stream is used. The current selection is then extended one page to the right.
blockMarkPageUp	If there is no selected text in the current view, then the current.block.defaultType setting is used to select a page of text starting at the current cursor position. If there is selected text in the current view, then the current selection is extended one page up from the current cursor position.
blockMarkPrevWord	First if the current selection is an element selection, then it is cleared. If there is no selected text in the current view, then the current.block.defaultType setting is used to select the current character. If the current.block.defaultType setting is element, then stream is used. The current selection is then extended to include the previous word.
blockMarkRectangle	If there is a rectangle selection within the current view, then the blockMarkRectangle action will extend the rectangle selection to the current cursor position. If there is no rectangle selection within the current view, then the blockMarkRectangle action will select the character at the current cursor position with the rectangle block type.
blockMarkRectangleAtMouse	The blockMarkRectangleAtMouse action will move the cursor to the mouse position, clear the current block selection (if any) and select the character at the new cursor position with the rectangular block type.

blockMarkRight	First if the current selection is an element selection, then it is cleared. If there is no selected text in the current view, then the current.block.defaultType setting is used to select the current character. If the current.block.defaultType setting is element, then stream is used. The current selection is then extended one character to the right. If the current selection type is character or stream and the cursor is at or beyond the end of the current line, then the selection is extended to include the first character on the next visible line.
blockMarkToMouse	If there is no selected text within the current view, then the blockMarkToMouse action will select text from the current position to the mouse position using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkToMouse action will extend the block selection to the mouse position.
blockMarkTop	If there is no selected text within the current view, then the blockMarkTop action will select text from the current position to the top of the file using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkTop action will extend the block selection to the top of the file.
blockMarkUp	If there is no selected text within the current view, then the blockMarkUp action will select text from the current cursor position to the corresponding position on the previous visible line using the current.block.defaultType setting for the block type. If there is some selected text within the current view, then the blockMarkUp action will extend the block selection to the previous visible line.
blockMarkWord	The blockMarkWord action selects the word at the current cursor location using current.block.defaultType for the block type. If the current.block.defaultType setting is element, then stream is used.
blockMarkWordAtMouse	The blockMarkWordAtMouse action selects the word at the mouse position using current.block.defaultType for the block type. If the current.block.defaultType setting is element, then stream is used.
blockMove	If the source and target views are not readonly and there is some visible text selected, then the blockMove action will move the selected text to the current cursor position. Otherwise, the blockMove action will do nothing. After the text has been moved, the moved text will be reselected.
blockOverlay	If the current view is not readonly and there is some visible text selected with either a rectangular or element selection, then the blockOverlay action will overlay the text at the current cursor position with the selected text. Otherwise, the blockOverlay action will do nothing. After the overlay, the new text will be reselected.
blockShiftLeft	If the current view is not readonly and there is some visible text selected with either a rectangular or element selection, then the blockShiftLeft action will shift the selected text one character position to the left. Otherwise, the blockShiftRight action will do nothing.

blockShiftRight	If the current view is not readonly and there is some visible text selected with either a rectangular or element selection, then the blockShiftRight action will shift the selected text one character position to the right. Otherwise, the blockShiftRight action will do nothing.
blockUnmark	If the block selection is contained within the current view, then the blockUnmark action will remove the selection.
blockUpperCase	If the current view is not readonly and there is some visible text selected, then the blockUpperCase action will upper case the selected text. Otherwise, the blockUpperCase action will do nothing.
bottom	The bottom action will move the cursor to the end of the last visible element in the document.
clearPrefix	The clearPrefix action will clear all of the text from the prefix area.
commandLine	The commandLine action will move the cursor from the text area to the command line.
compare	The compare action will invoke the compare dialog.
compareClear	The compareClear action will remove the compare information from the current document view.
compareNext	The compareNext action will move the cursor to the next mismatch in the current document view.
comparePrevious	The comparePrevious action will move the cursor to the previous mismatch in the current document view.
compareRefresh	The compareRefresh action will compare the current document view with the same file as it was last compared with.
copy	If the current view has some visible text selected, then the copy action will copy the selected text to the clipboard.
cursorToMouse	The cursorToMouse action moves the cursor to the current mouse position.
cut	If the current view is not readonly and there is some visible text selected, then the cut action will copy the selected text to the clipboard and delete the text from the document.
delete	The delete action will do one of the following: <ul style="list-style-type: none"> • If the cursor is beyond the end of the last visible line in the file, then nothing is done. • If the view is readonly, then nothing is done. • If the cursor is on a show line, then nothing is done. • If there is a stream selection in the current view, then the selected text is deleted. • If the cursor is not beyond the end of a line, then the character at the current cursor position is deleted. • If the cursor is beyond the end of a line, then the next visible line that is not a show line is joined with the current line.
deleteLine	If the current view is not readonly or the current line is a show element, then the deleteLine action will delete the current line.
deleteNextWord	If the current view is not readonly, then the deleteNextWord action will delete the next visible word in the document.

deletePrevWord	If the current view is not readonly, then the deletePrevWord action will delete the previous visible word in the document.
deleteToLineStart	If the current view is not readonly, then the deleteToLineStart action will delete the text from the current cursor position to the beginning of the line.
deleteWord	If the current view is not readonly, then the deleteWord action will delete the word at the current cursor position.
down	The down action will move the cursor to the next visible line.
duplicateLine	If the current view is not readonly and the current element is not a show line, then the duplicateLine action will copy the current line.
end	The end action will move the cursor to the end of the current line.
execCommand	The execCommand action will execute the text on the current line as an editor command.
expandHideAtMouse	The expandHideAtMouse action will toggle the expanded setting for the element under the mouse cursor between on and off . If the mouse cursor is at the top of the file, then the topExpanded setting will be toggled between on and off .
filterSelection	The filterSelection action will hide all of the lines in the document that do not contain the selected text.
find	The find action will move the cursor from the text area to the find line.
findAndReplace	The findAndReplace action will bring up the find and replace dialog.
findAndReplaceNext	If the current view is not readonly and the findText.findText parameter is not null, then then findAndReplaceNext command will replace the next instance of the text stored in the findText.findText parameter with the text stored in the findText.replaceText parameter.
findAndReplaceUp	If the current view is not readonly and the findText.findText parameter is not null, then then findAndReplaceNext command will replace the previous instance of the text stored in the findText.findText parameter with the text stored in the findText.replaceText parameter.
findBlockEnd	If the current view contains some visible selected text, then the findBlockEnd action will move the cursor to the end of the visible selected text.
findBlockStart	If the current view contains some visible selected text, then the findBlockStart action will move the cursor to the start of the visible selected text.
findLastChange	The findLastChange action will move the cursor to the last change made to the document.
findMark	The findMark action will move the cursor to the find mark line.
findNext	If the text stored in the findText.findText parameter is not a null string, then the findNext action will move the cursor to the next visible instance of the text stored in the findText.findText parameter.
findQuickMark	The findQuickMark action will move the cursor to the location of the quick mark that was set with the setQuickMark action.
findSelection	The findSelection action will move the cursor to the next visible instance of the selected text.

findUp	If the text stored in the findText.findText parameter is not a null string, then the findUp action will move the cursor to the previous visible instance of the text stored in the findText.findText parameter.
fonts	The fonts action allows the user to change the font that is used to display editor text.
get	The get action will bring up the get file dialog.
help	The help action will invoke the online help for the editor.
home	The home action will move the cursor to the beginning of the current line.
insertFileName	If the current view is not readonly and the current document is not an untitled document, then the insertFileName action will insert the name of the document at the current cursor position.
insertLeftBrace	If the current view is not readonly, then the insertLeftBrace action will insert a left brace at the current cursor position.
insertNot	If the current view is not readonly, then the insertNot action will insert a not character at the current cursor position.
insertRightBrace	If the current view is not readonly, then the insertRightBrace action will insert a right brace at the current cursor position.
insertTab	If the current view is not readonly, then the insertTab action will insert a tab character at the current cursor position.
join	The join action will join the current line with the next visible line that is not a show line.
keyRecorderPlay	The keyRecorderPlay action will play back the keystrokes that were recorded by the keyRecorderStart and keyRecorderStop actions.
keyRecorderStart	The keyRecorderStart action will start keystroke recording.
keyRecorderStop	The keyRecorderStop action will stop keystroke recording.
left	The left action will move the cursor one character position to the left.
locateLine	The locateLine action will move the cursor to the command line so that the user can enter a line number.
lowerCaseWord	If the current view is readonly, the lowerCaseWord action will change the word at the current cursor location to lower case.
nameMark	The nameMark action will move the cursor to the name mark line.
newLine	The newLine action will move the cursor to the beginning of the next visible line.
nextTabStop	The nextTabStop action will move the cursor to the next tab stop. Refer to current.tabs to determine the current tab stop settings.
nextWord	The nextWord action will move the cursor to the beginning of the next word.
nullAction	The nullAction action does nothing.
openLine	The openLine action will create a new line after the current line and move the cursor to the beginning of the new line.
pageDown	The pageDown action will move the cursor down one page.

pageLeft	The pageLeft action will move the cursor one page to the left.
pageRight	The pageRight action will move the cursor one page to the right.
pageUp	The pageUp action will move the cursor up one page.
paste	If the current view is not readonly then the paste action will copy the text from the clipboard to the current cursor location.
popupAtMouse	The popupAtMouse action will display the pop-up menu at the current mouse position.
preferences	Display the editor preferences dialog.
prefixBackSpace	The prefixBackSpace action will move the cursor position in the prefix area one character to the left and then delete the character at the current cursor position. If the cursor is already in column 1 of the prefix area then nothing will be done.
prefixDelete	The prefixDelete action will delete the character at the current cursor position in the prefix area.
prefixEnd	The prefixEnd action will move the current cursor position in the prefix area to the end of the text in the prefix area.
prefixHome	The prefixHome action will move the current cursor position in the prefix area to the beginning of the text in the prefix area.
prefixLeft	The prefixLeft action will move the current cursor position in the prefix area one character to the left. If the cursor is already in column 1, then nothing will be done.
prefixRight	The prefixRight action will move the current cursor position in the prefix area one character to the right. If the cursor is already at the end of the prefix text, then the cursor will be moved to column 1 of the text area.
prefixTruncate	The prefixTruncate action will delete the text from the current cursor position in the prefix area to the end of the prefix text.
prevTabStop	The prevTabStop action will move the cursor to the previous tab stop. Refer to current.tabs to determine the current tab stop settings.
prevWord	The prevWord action will move the cursor to the beginning of the previous word.
print	The print action will bring up the print dialog.
processPrefix	The processPrefix action will call the processPrefix command with the current baseProfile setting as the parameter. For example if the current base profile is seu , then the processPrefix action will issue the command processPrefix seu command. Note that the processPrefix command that is invoked is not necessarily the default processPrefix command. If the processPrefix command has been redefined for the current view, then the redefined processPrefix command will be called.
redo	The redo action will redo the last change that was undone with the undo action.
reload	The reload action will reload the document that is currently being edited.
rename	The rename action will allow the user to rename the current document.
right	The right action will move the cursor one character position to the right.

save	The save action will save the current document.
saveAs	The saveAs action will bring up the save as dialog.
scrollBottom	The scrollBottom action will scroll the current line to the bottom of the window.
scrollCenter	The scrollCenter action will scroll the current line to the middle of the window.
scrollTop	The scrollTop action will scroll the current line to the top of the window.
setParser	The setParser action will move the cursor to the set parser line.
setQuickMark	The setQuickMark action will set the quick mark at the current cursor location. Use the findQuickMark action to move the cursor back to the quick mark.
showAll	The showAll action will make all of the invisible lines visible.
split	The split action will split the current line at the current cursor position.
splitLine	The splitLine action will split the current line at the current cursor position and move the cursor to the beginning of the new line.
toggleCaseSensitive	The toggleCaseSensitive action will toggle the findText.asis parameter between off and on.
toggleInsert	The toggleInsert action will toggle the insert mode between insert and replace.
toggleKeyRecording	The toggleKeyRecording action will toggle the key recording mode between off and on.
toggleRegularExpression	The toggleRegularExpression action will toggle the findText.regularExpression parameter between off and on.
top	The top action will move the cursor to the beginning of the first visible line in the file.
truncate	If the current view is not readonly and the current line is not a show line, the truncate action will truncate the current line. Otherwise the truncate action will do nothing.
undo	The undo action will undo the last change to the document.
up	The up action will move the cursor to the previous visible line.
upperCaseWord	If the current view is readonly, the upperCaseWord action will change the word at the current cursor position to upper case.
windowBottom	The windowBottom action will move the cursor to the line at the bottom of the window.
windowTop	The windowTop action will move the cursor to the line at the top of the window.
wordEnd	The wordEnd action will move the cursor to the end of the word at the current cursor position.
wordStart	The wordStart action will move the cursor to the start of the word at the current cursor position.

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“baseProfile parameter” on page 66 parameter

“block.defaultType parameter” on page 67 parameter

“tabs parameter” on page 189 parameter

“expanded parameter” on page 94 parameter

“findText.asis parameter” on page 96 parameter

“findText.regularExpression parameter” on page 110 parameter

“findText.findText parameter” on page 106 parameter

“findText.replaceText parameter” on page 112 parameter

“topExpanded parameter” on page 193 parameter

brief base profile

The **brief** base profile has key assignments that should be familiar to brief users. Refer to the **updateProfile.baseProfile** parameter for other base profiles.

Key Settings

The key settings listed below are divided into key and action pairs. The first string (e.g. "a-a.t") indicates the key and the second string indicates the action (e.g. "blockMarkCharacter"). For a complete definition of how a key is defined refer to the **keyAction** parameter.

a-a.t	blockMarkCharacter
a-b.t.secondary	blockMarkCharacter
a-backSpace.t	deleteNextWord
a-c.t	blockMarkRectangle
a-d.t.secondary	deleteLine
a-enter.t.secondary	splitLine
a-f.t	insertFileName
a-f5.t	findUp
a-f6.t	findAndReplaceUp
a-g.t	locateLine
a-h.t	help
a-i.t	toggleInsert
a-j.t	findQuickMark
a-k.t	truncate
a-l.t	blockMarkElement
a-m.t.secondary	blockMarkCharacter
a-o.t	rename
a-p.t	print
a-pageDown.t	findBlockEnd
a-pageUp.t.secondary	findBlockStart
a-q.t.secondary	findQuickMark
a-r.t	get
a-s.t	split
a-t.t.secondary	findAndReplace

a-u.t	undo
a-w.t	save
a-y.t	findBlockStart
a-z.t	blockOverlay
backSpace.t	backSpace (page 222)
c-a.t	showAll
c-b.t	scrollBottom
c-backSpace.t	deletePrevWord
c-c.t	scrollCenter
c-d.t.secondary	pageDown
c-delete.t.secondary	truncate
c-e.t.secondary	pageUp
c-end.t	windowBottom
c-enter.t	openLine
c-f.t.secondary	find
c-f5.t	toggleCaseSensitive
c-f6.t	toggleRegularExpression
c-g.t.secondary	nullAction
c-h.t.secondary	nullAction
c-home.t	windowTop
c-i.t.secondary	nullAction
c-insert.t.secondary	copy
c-j.t	findLastChange
c-k.t	deleteToLineStart
c-l.t.secondary	locateLine
c-left.t	prevWord
c-m.t.secondary	nullAction
c-n.t.secondary	findNext
c-o.t.secondary	nullAction
c-p.t.secondary	print
c-pageDown.t	bottom
c-pageUp.t	top
c-q.t	setQuickMark
c-r.t.secondary	nullAction
c-right.t	nextWord
c-s-a.t.secondary	nullAction
c-s-b.t.secondary	nullAction
c-s-backSpace.t	deleteLine
c-s-c.t.secondary	nullAction
c-s-d.t.secondary	nullAction
c-s-e.t.secondary	nullAction
c-s-end.t	blockMarkBottom
c-s-f.t.secondary	findAndReplace
c-s-g.t.secondary	nullAction
c-s-h.t.secondary	nullAction
c-s-home.t	blockMarkTop
c-s-i.t.secondary	nullAction
c-s-j.t.secondary	nullAction
c-s-k.t.secondary	nullAction
c-s-l.t.secondary	nullAction
c-s-left.t	blockMarkPrevWord
c-s-m.t.secondary	nullAction
c-s-n.t	compareNext
c-s-o.t.secondary	nullAction
c-s-p.t	comparePrevious

c-s-pageDown.t
 c-s-pageUp.t
 c-s-q.t.secondary
 c-s-r.t
 c-s-right.t
 c-s-s.t.secondary
 c-s-t.t.secondary
 c-s-u.t.secondary
 c-s-v.t.secondary
 c-s-w.t.secondary
 c-s-x.t.secondary
 c-s-y.t.secondary
 c-s-z.t.secondary
 c-s.t.secondary
 c-t.t
 c-u.t
 c-v.t
 c-w.t
 c-x.t
 c-y.t
 c-z.t.secondary
 delete.t
 down.t
 end.t
 enter.t
 escape.t.secondary
 f1.t.secondary
 f10.t
 f5.t
 f6.t
 f7.t
 f8.t
 home.t
 insert.t.secondary
 left.t
 pageDown.t
 pageUp.t
 right.t
 s-delete.t.secondary
 s-down.t
 s-end.t
 s-enter.t.secondary
 s-f5.t
 s-f6.t
 s-f7.t
 s-home.t
 s-insert.t.secondary
 s-left.t
 s-pageDown.t
 s-pageUp.t
 s-right.t
 s-tab.t
 s-up.t
 tab.t
 up.t

blockMarkPageRight
 blockMarkPageLeft
 nullAction
 compareRefresh
 blockMarkNextWord
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 redo
 save
 scrollTop
 redo
 paste
 nullAction
 cut
 duplicateLine
 undo
 delete
 down
 end
 splitLine
 commandLine
 help
 commandLine
 find
 findAndReplace
 keyRecorderStart
 keyRecorderPlay
 home
 toggleInsert
 left
 pageDown
 pageUp
 right
 cut
 blockMarkDown
 blockMarkEnd
 splitLine
 findNext
 findAndReplaceNext
 keyRecorderStop
 blockMarkHome
 paste
 blockMarkLeft
 blockMarkPageDown
 blockMarkPageUp
 blockMarkRight
 prevTabStop
 blockMarkUp
 nextTabStop
 up

a-backSpace.p.secondary	undo
a-d.p	blockDelete
a-f5.p	findUp
a-f6.p	findAndReplaceUp
a-g.p	locateLine
a-h.p	help
a-i.p	toggleInsert
a-j.p	findQuickMark
a-k.p	blockUpperCase
a-o.p	rename
a-p.p	print
a-pageDown.p	findBlockEnd
a-pageUp.p.secondary	findBlockStart
a-q.p.secondary	findQuickMark
a-r.p	get
a-t.p.secondary	findAndReplace
a-u.p	undo
a-w.p	save
a-y.p	findBlockStart
backSpace.p	prefixBackSpace
c-a.p	showAll
c-b.p	scrollBottom
c-backSpace.p	deleteLine
c-c.p	scrollCenter
c-d.p.secondary	pageDown
c-delete.p	prefixTruncate
c-e.p.secondary	pageUp
c-end.p	windowBottom
c-enter.p	openLine
c-f.p.secondary	find
c-f5.p	toggleCaseSensitive
c-f6.p	toggleRegularExpression
c-g.p.secondary	nullAction
c-h.p.secondary	nullAction
c-home.p	windowTop
c-i.p.secondary	nullAction
c-j.p	findLastChange
c-k.p.secondary	nullAction
c-l.p.secondary	locateLine
c-m.p.secondary	nullAction
c-n.p.secondary	findNext
c-o.p.secondary	nullAction
c-p.p.secondary	print
c-pageDown.p	bottom
c-pageUp.p	top
c-q.p.secondary	nullAction
c-r.p.secondary	nullAction
c-s-a.p.secondary	nullAction
c-s-b.p.secondary	nullAction
c-s-c.p.secondary	nullAction
c-s-d.p.secondary	nullAction
c-s-e.p.secondary	nullAction
c-s-f.p.secondary	findAndReplace
c-s-g.p.secondary	nullAction
c-s-h.p.secondary	nullAction

c-s-i.p.secondary	nullAction
c-s-j.p.secondary	nullAction
c-s-k.p.secondary	nullAction
c-s-l.p.secondary	nullAction
c-s-m.p.secondary	nullAction
c-s-n.p	compareNext
c-s-o.p.secondary	nullAction
c-s-p.p	comparePrevious
c-s-q.p.secondary	nullAction
c-s-r.p	compareRefresh
c-s-s.p.secondary	nullAction
c-s-t.p.secondary	nullAction
c-s-u.p.secondary	nullAction
c-s-v.p.secondary	nullAction
c-s-w.p.secondary	nullAction
c-s-x.p.secondary	nullAction
c-s-y.p.secondary	nullAction
c-s-z.p.secondary	redo
c-s.p.secondary	save
c-t.p	scrollTop
c-u.p	redo
c-v.p.secondary	nullAction
c-w.p.secondary	nullAction
c-x.p	nullAction
c-y.p	duplicateLine
c-z.p.secondary	undo
delete.p	prefixDelete
down.p	down
end.p	prefixEnd
escape.p.secondary	commandLine
f1.p.secondary	help
f10.p	commandLine
f5.p	find
f6.p	findAndReplace
f7.p	keyRecorderStart
f8.p	keyRecorderPlay
home.p.secondary	prefixHome
insert.p.secondary	toggleInsert
left.p	prefixLeft
pageDown.p	pageDown
pageUp.p	pageUp
right.p	prefixRight
s-f5.p	findNext
s-f6.p	findAndReplaceNext
s-f7.p	keyRecorderStop
s-tab.p	prefixHome
tab.p	home
up.p	up
a-d.c	blockDelete
a-f5.c	findUp
a-f6.c	findAndReplaceUp
a-g.c	locateLine
a-h.c	help
a-i.c	blockLowerCase
a-j.c	findQuickMark

a-k.c	blockUpperCase
a-o.c	rename
a-p.c	print
a-pageDown.c	findBlockEnd
a-pageUp.c.secondary	findBlockStart
a-q.c.secondary	findQuickMark
a-r.c	get
a-t.c.secondary	findAndReplace
a-u.c	blockUnmark
a-w.c	save
a-y.c	findBlockStart
c-a.c	showAll
c-b.c	scrollBottom
c-backSpace.c	deleteLine
c-c.c	scrollCenter
c-d.c.secondary	pageDown
c-e.c.secondary	pageUp
c-end.c	windowBottom
c-enter.c	openLine
c-f.c.secondary	find
c-f5.c	toggleCaseSensitive
c-f6.c	toggleRegularExpression
c-home.c	windowTop
c-j.c	findLastChange
c-l.c.secondary	locateLine
c-n.c.secondary	findNext
c-p.c.secondary	print
c-pageDown.c	bottom
c-pageUp.c	top
c-s.c.secondary	save
c-t.c	scrollTop
c-u.c	redo
f1.c.secondary	help
f10.c	commandLine
f5.c	find
f6.c	findAndReplace
f7.c	keyRecorderStart
f8.c	keyRecorderPlay
pageDown.c	pageDown
pageUp.c	pageUp
s-f5.c	findNext
s-f6.c	findAndReplaceNext
s-f7.c	keyRecorderStop

Mouse Event Settings

The mouse event settings listed below are divided into mouse event and action pairs. The first string (e.g. "1-a-c-pressed.1.t.p.e") indicates the mouse event, and the second string indicates the action (e.g. "blockUnmark"). For a complete definition of how a mouse event is defined refer to the **mouseAction** parameter.

1-a-c-pressed.1.t	blockUnmark
1-a-c-s-pressed.1.t	blockUnmark
1-a-dragged.t	blockMarkToMouse
1-a-pressed.1.t	cursorToMouse

1-a-pressed.2.t
 1-c-dragged.t
 1-c-pressed.1.t
 1-c-pressed.2.t
 1-dragged.t
 1-pressed.1.t
 1-pressed.2.t
 1-s-dragged.t
 1-s-pressed.1.t
 2-dragged.t
 c-dragged.t
 dragged.t
 popup.t
 s-dragged.t
 1-a-c-pressed.1.p
 1-a-c-s-pressed.1.p
 1-a-dragged.p
 1-a-pressed.1.p
 1-a-pressed.2.p
 1-c-dragged.p
 1-c-pressed.1.p
 1-c-pressed.2.p
 1-dragged.p
 1-pressed.1.p
 1-pressed.2.p
 1-s-dragged.p
 1-s-pressed.1.p
 2-dragged.p
 c-dragged.p
 dragged.p
 popup.p
 s-dragged.p
 1-a-c-pressed.1.e
 1-a-c-s-pressed.1.e
 1-a-dragged.e
 1-a-pressed.1.e
 1-a-pressed.2.e
 1-c-dragged.e
 1-c-pressed.1.e
 1-c-pressed.2.e
 1-dragged.e
 1-pressed.1.e
 1-pressed.2.e
 1-pressed.3.e
 1-s-dragged.e
 1-s-pressed.1.e
 2-dragged.e
 c-dragged.e
 dragged.e
 popup.e
 s-dragged.e

blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkWordAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse
 blockUnmark
 blockUnmark
 blockMarkToMouse
 cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkWordAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse
 blockUnmark
 blockUnmark
 blockMarkToMouse
 cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 expandHideAtMouse
 expandHideAtMouse
 expandHideAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“keyAction parameter” on page 133 parameter

“mouseAction parameter” on page 151 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

epm base profile

The **epm** base profile has key assignments that should be familiar to epm users. Refer to the **updateProfile.baseProfile** parameter for other base profiles.

Key Settings

The key settings listed below are divided into key and action pairs. The first string (e.g. "a-b.t") indicates the key and the second string indicates the action (e.g. "blockMarkCharacter"). For a complete definition of how a key is defined refer to the **keyAction** parameter.

a-b.t	blockMarkRectangle
a-backSpace.t.secondary	undo
a-c.t	blockCopy
a-d.t	blockDelete
a-e.t	findBlockEnd
a-enter.t.secondary	splitLine
a-f.t	blockFill
a-i.t.secondary	blockLowerCase
a-j.t	join
a-k.t.secondary	blockUpperCase
a-l.t	blockMarkElement
a-m.t	blockMove
a-n.t	insertFileName
a-pageDown.t.secondary	findBlockEnd
a-pageUp.t.secondary	findBlockStart
a-q.t	findQuickMark
a-r.t.secondary	blockMarkRectangle
a-s.t	split
a-u.t	blockUnmark
a-w.t	blockMarkWord
a-y.t	findBlockStart
a-z.t	blockMarkCharacter
backSpace.t	backSpace (page 222)
c-0.t	insertRightBrace
c-6.t	insertNot
c-9.t	insertLeftBrace
c-a.t	showAll
c-b.t	findMark
c-backSpace.t.secondary	deleteLine

c-c.t
 c-d.t
 c-delete.t.secondary
 c-e.t
 c-end.t.secondary
 c-enter.t.secondary
 c-f.t
 c-f1.t
 c-f2.t
 c-f3.t
 c-f4.t
 c-f5.t
 c-f6.t
 c-f7.t
 c-f8.t
 c-g.t.secondary
 c-h.t.secondary
 c-home.t.secondary
 c-i.t
 c-insert.t.secondary
 c-j.t
 c-k.t
 c-l.t
 c-left.t.secondary
 c-m.t
 c-n.t.secondary
 c-o.t.secondary
 c-p.t
 c-pageDown.t.secondary
 c-pageUp.t.secondary
 c-q.t
 c-r.t
 c-right.t.secondary
 c-s-a.t.secondary
 c-s-b.t.secondary
 c-s-backSpace.t
 c-s-c.t.secondary
 c-s-d.t.secondary
 c-s-e.t.secondary
 c-s-end.t
 c-s-f.t
 c-s-g.t.secondary
 c-s-h.t.secondary
 c-s-home.t
 c-s-i.t.secondary
 c-s-j.t.secondary
 c-s-k.t.secondary
 c-s-l.t.secondary
 c-s-left.t
 c-s-m.t.secondary
 c-s-n.t
 c-s-o.t.secondary
 c-s-p.t
 c-s-pageDown.t
 c-s-pageUp.t

copy
 deleteWord
 truncate
 truncate
 bottom
 splitLine
 findNext
 upperCaseWord
 lowerCaseWord
 blockUpperCase
 blockLowerCase
 wordStart
 wordEnd
 blockShiftLeft
 blockShiftRight
 nullAction
 nullAction
 top
 commandLine
 copy
 findLastChange
 duplicateLine
 execCommand
 prevWord
 nameMark
 findNext
 nullAction
 print
 pageRight
 pageLeft
 setQuickMark
 toggleKeyRecording
 nextWord
 nullAction
 nullAction
 nullAction
 deleteLine
 nullAction
 nullAction
 nullAction
 nullAction
 bottom
 findAndReplace
 nullAction
 nullAction
 top
 nullAction
 nullAction
 nullAction
 nullAction
 prevWord
 nullAction
 compareNext
 nullAction
 comparePrevious
 blockMarkPageRight
 blockMarkPageLeft

c-s-q.t.secondary	nullAction
c-s-r.t	compareRefresh
c-s-right.t	nextWord
c-s-s.t.secondary	nullAction
c-s-t.t.secondary	nullAction
c-s-u.t.secondary	nullAction
c-s-v.t.secondary	nullAction
c-s-w.t.secondary	nullAction
c-s-x.t.secondary	nullAction
c-s-y.t.secondary	nullAction
c-s-z.t	redo
c-s.t	find
c-t.t	keyRecorderPlay
c-tab.t	insertTab
c-u.t.secondary	undo
c-v.t	paste
c-w.t	nullAction
c-x.t	cut
c-y.t.secondary	duplicateLine
c-z.t.secondary	undo
delete.t	delete
down.t	down
end.t	end
enter.t	splitLine
escape.t.secondary	commandLine
f1.t	help
f2.t	save
f3.t	rename
f7.t.secondary	blockShiftLeft
f8.t.secondary	blockShiftRight
f9.t	undo
home.t	home
insert.t	toggleInsert
left.t	left
pageDown.t.secondary	pageDown
pageUp.t.secondary	pageUp
right.t	right
s-delete.t.secondary	cut
s-down.t	blockMarkDown
s-end.t	blockMarkEnd
s-enter.t.secondary	splitLine
s-f1.t	pageLeft
s-f2.t	pageRight
s-f3.t	pageUp
s-f4.t	pageDown
s-home.t	blockMarkHome
s-insert.t.secondary	paste
s-left.t	blockMarkLeft
s-pageDown.t	blockMarkPageDown
s-pageUp.t	blockMarkPageUp
s-right.t	blockMarkRight
s-tab.t	prevTabStop
s-up.t	blockMarkUp
tab.t	nextTabStop
up.t	up

- a-backSpace.p.secondary
- a-d.p
- a-e.p
- a-f.p
- a-i.p.secondary
- a-k.p.secondary
- a-pageDown.p.secondary
- a-pageUp.p.secondary
- a-q.p
- a-u.p
- a-y.p
- backSpace.p
- c-a.p
- c-b.p
- c-backSpace.p
- c-c.p.secondary
- c-d.p.secondary
- c-delete.p
- c-e.p.secondary
- c-end.p.secondary
- c-enter.p
- c-f.p
- c-f3.p
- c-f4.p
- c-g.p.secondary
- c-h.p.secondary
- c-home.p.secondary
- c-i.p
- c-j.p
- c-k.p
- c-l.p
- c-m.p.secondary
- c-n.p.secondary
- c-o.p.secondary
- c-p.p
- c-pageDown.p.secondary
- c-pageUp.p.secondary
- c-q.p.secondary
- c-r.p
- c-s-a.p.secondary
- c-s-b.p.secondary
- c-s-c.p.secondary
- c-s-d.p.secondary
- c-s-e.p.secondary
- c-s-end.p
- c-s-f.p
- c-s-g.p.secondary
- c-s-h.p.secondary
- c-s-home.p
- c-s-i.p.secondary
- c-s-j.p.secondary
- c-s-k.p.secondary
- c-s-l.p.secondary
- c-s-m.p.secondary
- c-s-n.p

- undo
- blockDelete
- findBlockEnd
- blockFill
- blockLowerCase
- blockUpperCase
- findBlockEnd
- findBlockStart
- findQuickMark
- blockUnmark
- findBlockStart
- prefixBackSpace
- showAll
- findMark
- deleteLine
- nullAction
- nullAction
- prefixTruncate
- nullAction
- bottom
- openLine
- findNext
- blockUpperCase
- blockLowerCase
- nullAction
- nullAction
- top
- commandLine
- findLastChange
- duplicateLine
- locateLine
- nullAction
- findNext
- nullAction
- print
- pageRight
- pageLeft
- nullAction
- toggleKeyRecording
- nullAction
- nullAction
- nullAction
- nullAction
- bottom
- findAndReplace
- nullAction
- nullAction
- top
- nullAction
- nullAction
- nullAction
- nullAction
- compareNext

c-s-o.p.secondary	nullAction
c-s-p.p	comparePrevious
c-s-q.p.secondary	nullAction
c-s-r.p	compareRefresh
c-s-s.p.secondary	nullAction
c-s-t.p.secondary	nullAction
c-s-u.p.secondary	nullAction
c-s-v.p.secondary	nullAction
c-s-w.p.secondary	nullAction
c-s-x.p.secondary	nullAction
c-s-y.p.secondary	nullAction
c-s-z.p	redo
c-s.p	find
c-t.p	keyRecorderPlay
c-u.p.secondary	undo
c-v.p.secondary	nullAction
c-w.p.secondary	nullAction
c-x.p	nullAction
c-y.p.secondary	duplicateLine
c-z.p.secondary	undo
delete.p	prefixDelete
down.p	down
end.p	prefixEnd
escape.p.secondary	commandLine
f1.p	help
f2.p	save
f3.p	rename
f7.p	blockShiftLeft
f8.p	blockShiftRight
f9.p	undo
home.p.secondary	prefixHome
insert.p	toggleInsert
left.p	prefixLeft
pageDown.p.secondary	pageDown
pageUp.p.secondary	pageUp
right.p	prefixRight
s-f1.p	pageLeft
s-f2.p	pageRight
s-f3.p	pageUp
s-f4.p	pageDown
s-tab.p	prefixHome
tab.p	home
up.p	up
a-d.c	blockDelete
a-e.c	findBlockEnd
a-f.c	blockFill
a-i.c.secondary	blockLowerCase
a-k.c.secondary	blockUpperCase
a-pageDown.c.secondary	findBlockEnd
a-pageUp.c.secondary	findBlockStart
a-q.c	findQuickMark
a-u.c	blockUnmark
a-y.c	findBlockStart
c-a.c	showAll
c-b.c	findMark

c-backSpace.c	deleteLine
c-end.c.secondary	bottom
c-enter.c	openLine
c-f.c	findNext
c-f3.c	blockUpperCase
c-f4.c	blockLowerCase
c-home.c.secondary	top
c-j.c	findLastChange
c-l.c	locateLine
c-n.c.secondary	findNext
c-p.c	print
c-pageDown.c.secondary	pageRight
c-pageUp.c.secondary	pageLeft
c-r.c	toggleKeyRecording
c-s-end.c	bottom
c-s-home.c	top
c-s.c	find
c-t.c	keyRecorderPlay
c-u.c	findUp
f1.c	help
f2.c	save
f3.c	rename
f7.c	blockShiftLeft
f8.c	blockShiftRight
f9.c	undo
pageDown.c.secondary	pageDown
pageUp.c.secondary	pageUp
s-f1.c	pageLeft
s-f2.c	pageRight
s-f3.c	pageUp
s-f4.c	pageDown

Mouse Event Settings

The mouse event settings listed below are divided into mouse event and action pairs. The first string (e.g. "1-a-c-pressed.1.t.p.e") indicates the mouse event and the second string indicates the action (e.g. "blockUnmark"). For a complete definition of how a mouse event is defined refer to the **mouseAction** parameter.

1-a-c-pressed.1.t	blockUnmark
1-a-c-s-pressed.1.t	blockUnmark
1-a-dragged.t	blockMarkToMouse
1-a-pressed.1.t	cursorToMouse
1-a-pressed.2.t	blockMarkRectangleAtMouse
1-c-dragged.t	blockMarkToMouse
1-c-pressed.1.t	cursorToMouse
1-c-pressed.2.t	blockMarkElementAtMouse
1-dragged.t	blockMarkToMouse
1-pressed.1.t	cursorToMouse
1-pressed.2.t	blockMarkWordAtMouse
1-s-dragged.t	blockMarkToMouse
1-s-pressed.1.t	blockMarkToMouse
2-dragged.t	blockMarkToMouse
c-dragged.t	blockMarkToMouse
dragged.t	blockMarkToMouse

popup.t	popupAtMouse
s-dragged.t	blockMarkToMouse
1-a-c-pressed.1.p	blockUnmark
1-a-c-s-pressed.1.p	blockUnmark
1-a-dragged.p	blockMarkToMouse
1-a-pressed.1.p	cursorToMouse
1-a-pressed.2.p	blockMarkRectangleAtMouse
1-c-dragged.p	blockMarkToMouse
1-c-pressed.1.p	cursorToMouse
1-c-pressed.2.p	blockMarkElementAtMouse
1-dragged.p	blockMarkToMouse
1-pressed.1.p	cursorToMouse
1-pressed.2.p	blockMarkWordAtMouse
1-s-dragged.p	blockMarkToMouse
1-s-pressed.1.p	blockMarkToMouse
2-dragged.p	blockMarkToMouse
c-dragged.p	blockMarkToMouse
dragged.p	blockMarkToMouse
popup.p	popupAtMouse
s-dragged.p	blockMarkToMouse
1-a-c-pressed.1.e	blockUnmark
1-a-c-s-pressed.1.e	blockUnmark
1-a-dragged.e	blockMarkToMouse
1-a-pressed.1.e	cursorToMouse
1-a-pressed.2.e	blockMarkRectangleAtMouse
1-c-dragged.e	blockMarkToMouse
1-c-pressed.1.e	cursorToMouse
1-c-pressed.2.e	blockMarkElementAtMouse
1-dragged.e	blockMarkToMouse
1-pressed.1.e	expandHideAtMouse
1-pressed.2.e	expandHideAtMouse

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“keyAction parameter” on page 133 parameter

“mouseAction parameter” on page 151 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

ispf base profile

The **ispf** base profile has key assignments and prefix commands that should be familiar to ispf users. Refer to the **updateProfile.baseProfile** parameter for other base profiles.

Key Settings

The key settings listed below are divided into key and action pairs. The first string (e.g. “a-b.t”) indicates the key and the second string indicates the action (e.g.

"blockMarkCharacter"). For a complete definition of how a key is defined refer to the **keyAction** parameter.

a-b.t	blockMarkCharacter
a-backSpace.t.secondary	undo
a-c.t	blockCopy
a-d.t	blockDelete
a-i.t	blockLowerCase
a-j.t	join
a-k.t	blockUpperCase
a-l.t	blockMarkElement
a-m.t	blockMove
a-pageDown.t	findBlockEnd
a-pageUp.t	findBlockStart
a-q.t	findQuickMark
a-r.t	blockMarkRectangle
a-s.t	split
a-u.t	blockUnmark
a-z.t	blockOverlay
backSpace.t	backSpace (page 222)
c-a.t	showAll
c-b.t.secondary	nullAction
c-backSpace.t	deleteLine
c-c.t	copy
c-d.t.secondary	nullAction
c-delete.t	truncate
c-e.t.secondary	nullAction
c-end.t	bottom
c-enter.t	openLine
c-f.t	find
c-g.t.secondary	nullAction
c-h.t.secondary	nullAction
c-home.t	top
c-i.t.secondary	nullAction
c-insert.t.secondary	copy
c-j.t	findLastChange
c-k.t.secondary	nullAction
c-l.t	locateLine
c-left.t	prevWord
c-m.t.secondary	nullAction
c-n.t	findNext
c-o.t.secondary	nullAction
c-p.t	print
c-pageDown.t	pageRight
c-pageUp.t	pageLeft
c-q.t	setQuickMark
c-r.t.secondary	nullAction
c-right.t	nextWord
c-s-a.t.secondary	nullAction
c-s-b.t.secondary	nullAction
c-s-c.t.secondary	nullAction
c-s-d.t.secondary	nullAction
c-s-e.t.secondary	nullAction
c-s-end.t	blockMarkBottom
c-s-f.t	findAndReplace
c-s-g.t.secondary	nullAction

c-s-h.t.secondary	nullAction
c-s-home.t	blockMarkTop
c-s-i.t.secondary	nullAction
c-s-j.t.secondary	nullAction
c-s-k.t.secondary	nullAction
c-s-l.t.secondary	nullAction
c-s-left.t	blockMarkPrevWord
c-s-m.t.secondary	nullAction
c-s-n.t	compareNext
c-s-o.t.secondary	nullAction
c-s-p.t	comparePrevious
c-s-pageDown.t	blockMarkPageRight
c-s-pageUp.t	blockMarkPageLeft
c-s-q.t.secondary	nullAction
c-s-r.t	compareRefresh
c-s-right.t	blockMarkNextWord
c-s-s.t.secondary	nullAction
c-s-t.t.secondary	nullAction
c-s-u.t.secondary	nullAction
c-s-v.t.secondary	nullAction
c-s-w.t.secondary	nullAction
c-s-x.t.secondary	nullAction
c-s-y.t.secondary	nullAction
c-s-z.t	redo
c-s.t	save
c-t.t.secondary	nullAction
c-u.t	findUp
c-v.t	paste
c-w.t	nullAction
c-x.t	cut
c-y.t	duplicateLine
c-z.t	undo
delete.t	delete
down.t	down
end.t	end
enter.t	splitLine
escape.t	commandLine
f1.t	help
f7.t	blockShiftLeft
f8.t	blockShiftRight
home.t	home
insert.t	toggleInsert
left.t	left
pageDown.t	pageDown
pageUp.t	pageUp
right.t	right
s-delete.t.secondary	cut
s-down.t	blockMarkDown
s-end.t	blockMarkEnd
s-enter.t	newLine
s-home.t	blockMarkHome
s-insert.t.secondary	paste
s-left.t	blockMarkLeft
s-pageDown.t	blockMarkPageDown
s-pageUp.t	blockMarkPageUp

s-right.t
 s-tab.t
 s-up.t
 tab.t
 up.t
 a-backSpace.p.secondary
 a-d.p
 a-i.p
 a-k.p
 a-pageDown.p
 a-pageUp.p
 a-q.p
 a-u.p
 backSpace.p
 c-a.p
 c-b.p.secondary
 c-backSpace.p
 c-c.p.secondary
 c-d.p.secondary
 c-delete.p
 c-e.p.secondary
 c-end.p
 c-enter.p
 c-f.p
 c-g.p.secondary
 c-h.p.secondary
 c-home.p
 c-i.p.secondary
 c-j.p
 c-k.p.secondary
 c-l.p
 c-m.p.secondary
 c-n.p
 c-o.p.secondary
 c-p.p
 c-pageDown.p
 c-pageUp.p
 c-q.p.secondary
 c-r.p.secondary
 c-s-a.p.secondary
 c-s-b.p.secondary
 c-s-c.p.secondary
 c-s-d.p.secondary
 c-s-e.p.secondary
 c-s-f.p
 c-s-g.p.secondary
 c-s-h.p.secondary
 c-s-i.p.secondary
 c-s-j.p.secondary
 c-s-k.p.secondary
 c-s-l.p.secondary
 c-s-m.p.secondary
 c-s-n.p
 c-s-o.p.secondary
 c-s-p.p

blockMarkRight
 prevTabStop
 blockMarkUp
 nextTabStop
 up
 undo
 blockDelete
 blockLowerCase
 blockUpperCase
 findBlockEnd
 findBlockStart
 findQuickMark
 blockUnmark
 prefixBackSpace
 showAll
 nullAction
 deleteLine
 nullAction
 nullAction
 prefixTruncate
 nullAction
 bottom
 openLine
 find
 nullAction
 nullAction
 top
 nullAction
 findLastChange
 nullAction
 locateLine
 nullAction
 findNext
 nullAction
 print
 pageRight
 pageLeft
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 findAndReplace
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 compareNext
 nullAction
 comparePrevious

c-s-q.p.secondary
 c-s-r.p
 c-s-s.p.secondary
 c-s-t.p.secondary
 c-s-u.p.secondary
 c-s-v.p.secondary
 c-s-w.p.secondary
 c-s-x.p.secondary
 c-s-y.p.secondary
 c-s-z.p
 c-s.p
 c-t.p.secondary
 c-u.p
 c-v.p.secondary
 c-w.p.secondary
 c-x.p
 c-y.p
 c-z.p
 delete.p
 down.p
 end.p
 enter.p
 escape.p
 f1.p
 f7.p
 f8.p
 home.p.secondary
 insert.p
 left.p
 pageDown.p
 pageUp.p
 right.p
 s-tab.p
 tab.p
 up.p
 a-d.c
 a-i.c
 a-k.c
 a-pageDown.c
 a-pageUp.c
 a-q.c
 a-u.c
 c-a.c
 c-backSpace.c
 c-end.c
 c-enter.c
 c-f.c
 c-home.c
 c-j.c
 c-l.c
 c-n.c
 c-p.c
 c-pageDown.c
 c-pageUp.c
 c-s.c

nullAction
 compareRefresh
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 redo
 save
 nullAction
 findUp
 nullAction
 nullAction
 nullAction
 duplicateLine
 undo
 prefixDelete
 down
 prefixEnd
 processPrefix
 commandLine
 help
 blockShiftLeft
 blockShiftRight
 prefixHome
 toggleInsert
 prefixLeft
 pageDown
 pageUp
 prefixRight
 prefixHome
 home
 up
 blockDelete
 blockLowerCase
 blockUpperCase
 findBlockEnd
 findBlockStart
 findQuickMark
 blockUnmark
 showAll
 deleteLine
 bottom
 openLine
 find
 top
 findLastChange
 locateLine
 findNext
 print
 pageRight
 pageLeft
 save

c-u.c
f1.c
f7.c
f8.c
pageDown.c
pageUp.c

findUp
help
blockShiftLeft
blockShiftRight
pageDown
pageUp

Mouse Event Settings

The mouse event settings listed below are divided into mouse event and action pairs. The first string (e.g. "1-a-c-pressed.1.t.p.e") indicates the mouse event and the second string indicates the action (e.g. "blockUnmark"). For a complete definition of how a mouse event is defined refer to the **mouseAction** parameter.

1-a-c-pressed.1.t	blockUnmark
1-a-c-s-pressed.1.t	blockUnmark
1-a-dragged.t	blockMarkToMouse
1-a-pressed.1.t	cursorToMouse
1-a-pressed.2.t	blockMarkRectangleAtMouse
1-c-dragged.t	blockMarkToMouse
1-c-pressed.1.t	cursorToMouse
1-c-pressed.2.t	blockMarkElementAtMouse
1-dragged.t	blockMarkToMouse
1-pressed.1.t	cursorToMouse
1-pressed.2.t	blockMarkWordAtMouse
1-s-dragged.t	blockMarkToMouse
1-s-pressed.1.t	blockMarkToMouse
2-dragged.t	blockMarkToMouse
c-dragged.t	blockMarkToMouse
dragged.t	blockMarkToMouse
popup.t	popupAtMouse
s-dragged.t	blockMarkToMouse
1-a-c-pressed.1.p	blockUnmark
1-a-c-s-pressed.1.p	blockUnmark
1-a-dragged.p	blockMarkToMouse
1-a-pressed.1.p	cursorToMouse
1-a-pressed.2.p	blockMarkRectangleAtMouse
1-c-dragged.p	blockMarkToMouse
1-c-pressed.1.p	cursorToMouse
1-c-pressed.2.p	blockMarkElementAtMouse
1-dragged.p	blockMarkToMouse
1-pressed.1.p	cursorToMouse
1-pressed.2.p	blockMarkWordAtMouse
1-s-dragged.p	blockMarkToMouse
1-s-pressed.1.p	blockMarkToMouse
2-dragged.p	blockMarkToMouse
c-dragged.p	blockMarkToMouse
dragged.p	blockMarkToMouse
popup.p	popupAtMouse
s-dragged.p	blockMarkToMouse
1-a-c-pressed.1.e	blockUnmark
1-a-c-s-pressed.1.e	blockUnmark
1-a-dragged.e	blockMarkToMouse

1-a-pressed.1.e
 1-a-pressed.2.e
 1-c-dragged.e
 1-c-pressed.1.e
 1-c-pressed.2.e
 1-dragged.e
 1-pressed.1.e
 1-pressed.2.e
 1-pressed.3.e
 1-s-dragged.e
 1-s-pressed.1.e
 2-dragged.e
 c-dragged.e
 dragged.e
 popup.e
 s-dragged.e

cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 expandHideAtMouse
 expandHideAtMouse
 expandHideAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse

Prefix Commands

(

Enter (in the prefix area to cause the editor to shift the specified line one character to the left.

(*n*

Enter (*n* in the prefix area to cause the editor to shift the specified line *n* characters to the left.

((

Enter ((in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the left. Only the visible lines will be shifted.

((*n*

Enter ((*n* in the prefix area of two different lines in the document to delimit a block of text that you want to shift *n* characters to the left. Only the visible lines will be shifted.

)

Enter) in the prefix area to cause the editor to shift the specified line one character to the right.

)*n*

Enter)*n* in the prefix area to cause the editor to shift the specified line *n* characters to the right.

))

Enter)) in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the right. Only the visible lines will be shifted.

))*n*

Enter))*n* in the prefix area of two different lines in the document to delimit a block of text that you want to shift *n* characters to the right. Only the visible lines will be shifted.

/

Enter / in the prefix area to make the specified line the current line.

<

Enter < in the prefix area to cause the editor to shift the specified line one character to the left. If there is not at least one blank at the start of the line, then nothing is done.

<code><n</code>	Enter <code><n</code> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the left. If there are less than <i>n</i> spaces at the start of the line, then only the spaces are deleted.
<code><<</code>	Enter <code><<</code> in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the left. Only the visible lines will be shifted. Lines that do not have at least one space at the start of the line will be unaffected.
<code><<n</code>	Enter <code>>>n</code> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the left. Only the visible lines will be shifted. Lines that have less than <i>n</i> spaces at the start of the line will only have those spaces deleted.
<code>></code>	Enter <code>></code> in the prefix area to cause the editor to shift the specified line one character to the right. If the line cannot be shifted without truncating a non blank character, then nothing is done.
<code>>n</code>	Enter <code>>n</code> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the right. If the line cannot be shifted <i>n</i> characters without truncating a non blank character, then the line will only be shifted as far to the right as it can be without truncating a non blank character.
<code>>></code>	Enter <code>>></code> in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the right. Only the visible lines will be shifted. Lines that cannot be shifted without truncating a non blank character will be unaffected.
<code>>>n</code>	Enter <code>>>n</code> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the right. Only the visible lines will be shifted. Lines that cannot be shifted without truncating a non blank character will only be shifted as far to the right as they can be without truncating a non blank character.
<code>A</code>	Enter <code>A</code> in the prefix area to cause the editor to copy or move lines after the specified line. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: <code>C</code> , <code>CC</code> , <code>M</code> , or <code>MM</code> .
<code>B</code>	Enter <code>B</code> in the prefix area to cause the editor to copy or move lines before the specified line. The source lines and type of operation (move or copy) are determined by the first occurrence of one of the following commands: <code>C</code> , <code>CC</code> , <code>M</code> , or <code>MM</code> .

C	Enter C in the prefix area to indicate that the specified line is to be the source for a copy operation. The target location for the copy is determined by the first occurrence of one of an A , B , O , or OO command.
CC	Enter CC in the prefix area of two different lines in the document. The CC commands will delimit the source for a copy operation. Only visible lines are included in the source. The target location for the copy is determined by the first occurrence of one of an A , B , O , or OO command.
D	Enter D in the prefix area to cause the editor to delete the specified line.
Dn	Enter Dn in the prefix area to cause the editor to delete the specified line and the following $n - 1$ visible lines.
DD	Enter DD in the prefix area of two different lines in the document to delimit a block of text that you want deleted. Only the visible lines will be deleted.
F	Enter F in the prefix area of an excluded block header to show the first line in the excluded block. Lines can be excluded with the X , Xn , and XX commands.
Fn	Enter Fn in the prefix area of an excluded block header to show the first n lines in the excluded block. Lines can be excluded with the X , Xn , and XX commands.
I	Enter I in the prefix area to cause the editor to insert a new line after the specified line.
In	Enter In in the prefix area to cause the editor to insert n new lines after the specified line. n is a positive integer.
L	Enter L in the prefix area of an excluded block header to show the last line in the excluded block. Lines can be excluded with the X , Xn , and XX commands.
Ln	Enter Ln in the prefix area of an excluded block header to show the last n lines in the excluded block. Lines can be excluded with the X , Xn , and XX commands.
LC	Enter LC in the prefix area to cause the editor to change the specified line to lower case.
LCn	Enter LCn in the prefix area to cause the editor to change the specified line and the following $n-1$ visible lines to lower case.
LCC	Enter LCC in the prefix area of two different lines in the document to delimit a block of text that you want to change to lower case. Only the visible lines will be changed.
M	Enter M in the prefix area to indicate that the specified line is to be the source for a move operation. The target location for the move is determined by the first occurrence of one of an A , B , O , or OO command.

MM	Enter M in the prefix area to indicate that the specified line is to be the source for a move operation. The target location for the move is determined by the first occurrence of one of an A , B , O , or OO command.
O	Enter O in the prefix area to cause the editor to overlay the specified line. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , CC , M , or MM .
OO	Enter OO in the prefix area of two different lines in the document. The OO commands will delimit the overlay target for a move or copy operation. Only the visible lines will be overlaid. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , CC , M , or MM .
R	Enter R in the prefix area to cause the editor to duplicate the specified line.
Rn	Enter Rn in the prefix area to cause the editor to duplicate the specified line <i>n</i> times.
RR	Enter RR in the prefix area of two different lines in the document. The RR commands will delimit a block of text that you want duplicated. Only the visible lines will be duplicated.
RRn	Enter RRn in the prefix area of two different lines in the document. The RRn commands will delimit a block of text that you want duplicated <i>n</i> times. Only the visible lines will be duplicated.
UC	Enter UC in the prefix area to cause the editor to change the specified line to upper case.
UCn	Enter UCn in the prefix area to cause the editor to change the specified line and the following <i>n</i> -1 visible lines to upper case.
UCC	Enter UCC in the prefix area of two different lines in the document to delimit a block of text that you want to change to upper case. Only the visible lines will be changed.
X	Enter X in the prefix area to cause the editor to exclude the specified line. The line can be reshown with the L , Ln , F , or Fn commands.
Xn	Enter Xn in the prefix area to cause the editor to exclude the specified line and the following <i>n</i> - 1 visible lines. The line can be reshown with the L , Ln , F , or Fn commands.
XX	Enter XX in the prefix area of two different lines in the document to delimit a block of text that you want to exclude. The line can be reshown with the L , Ln , F , or Fn commands.

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“keyAction parameter” on page 133 parameter

“mouseAction parameter” on page 151 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

Ipex base profile

The **Ipex** base profile is the default base profile. Refer to the **updateProfile.baseProfile** parameter for other base profiles.

Key Settings

The key settings listed below are divided into key and action pairs. The first string (e.g. "a-b.t") indicates the key and the second string indicates the action (e.g. "blockMarkCharacter"). For a complete definition of how a key is defined refer to the **keyAction** parameter.

a-b.t	blockMarkCharacter
a-backSpace.t.secondary	undo
a-c.t	blockCopy
a-d.t	blockDelete
a-i.t	blockLowerCase
a-j.t	join
a-k.t	blockUpperCase
a-l.t	blockMarkElement
a-m.t	blockMove
a-pageDown.t	findBlockEnd
a-pageUp.t	findBlockStart
a-q.t	findQuickMark
a-r.t	blockMarkRectangle
a-s.t	split
a-u.t	blockUnmark
a-z.t	blockOverlay
backSpace.t	backSpace (page 222)
c-a.t	showAll
c-b.t.secondary	nullAction
c-backSpace.t	deleteLine
c-c.t	copy
c-d.t.secondary	nullAction
c-delete.t	truncate
c-e.t.secondary	nullAction
c-end.t	bottom
c-enter.t	openLine
c-f.t	find
c-g.t.secondary	nullAction
c-h.t.secondary	nullAction

c-home.t
 c-i.t.secondary
 c-insert.t.secondary
 c-j.t
 c-k.t.secondary
 c-l.t
 c-left.t
 c-m.t.secondary
 c-n.t
 c-o.t.secondary
 c-p.t
 c-pageDown.t
 c-pageUp.t
 c-q.t
 c-r.t.secondary
 c-right.t
 c-s-a.t.secondary
 c-s-b.t.secondary
 c-s-c.t.secondary
 c-s-d.t.secondary
 c-s-e.t.secondary
 c-s-end.t
 c-s-f.t
 c-s-g.t.secondary
 c-s-h.t.secondary
 c-s-home.t
 c-s-i.t.secondary
 c-s-j.t.secondary
 c-s-k.t.secondary
 c-s-l.t.secondary
 c-s-left.t
 c-s-m.t.secondary
 c-s-n.t
 c-s-o.t.secondary
 c-s-p.t
 c-s-pageDown.t
 c-s-pageUp.t
 c-s-q.t.secondary
 c-s-r.t
 c-s-right.t
 c-s-s.t.secondary
 c-s-t.t.secondary
 c-s-u.t.secondary
 c-s-v.t.secondary
 c-s-w.t.secondary
 c-s-x.t.secondary
 c-s-y.t.secondary
 c-s-z.t
 c-s.t
 c-t.t.secondary
 c-u.t
 c-v.t
 c-w.t
 c-x.t
 c-y.t

top
 nullAction
 copy
 findLastChange
 nullAction
 locateLine
 prevWord
 nullAction
 findNext
 nullAction
 print
 pageRight
 pageLeft
 setQuickMark
 nullAction
 nextWord
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 blockMarkBottom
 findAndReplace
 nullAction
 nullAction
 blockMarkTop
 nullAction
 nullAction
 nullAction
 nullAction
 blockMarkPrevWord
 nullAction
 compareNext
 nullAction
 comparePrevious
 blockMarkPageRight
 blockMarkPageLeft
 nullAction
 compareRefresh
 blockMarkNextWord
 nullAction
 nullAction
 nullAction
 nullAction
 nullAction
 redo
 save
 nullAction
 findUp
 paste
 nullAction
 cut
 duplicateLine

c-z.t	undo
delete.t	delete
down.t	down
end.t	end
enter.t	splitLine
escape.t	commandLine
f1.t	help
f7.t	blockShiftLeft
f8.t	blockShiftRight
home.t	home
insert.t	toggleInsert
left.t	left
pageDown.t	pageDown
pageUp.t	pageUp
right.t	right
s-delete.t.secondary	cut
s-down.t	blockMarkDown
s-end.t	blockMarkEnd
s-enter.t	newLine
s-home.t	blockMarkHome
s-insert.t.secondary	paste
s-left.t	blockMarkLeft
s-pageDown.t	blockMarkPageDown
s-pageUp.t	blockMarkPageUp
s-right.t	blockMarkRight
s-tab.t	prevTabStop
s-up.t	blockMarkUp
tab.t	nextTabStop
up.t	up
a-backSpace.p.secondary	undo
a-d.p	blockDelete
a-i.p	blockLowerCase
a-k.p	blockUpperCase
a-pageDown.p	findBlockEnd
a-pageUp.p	findBlockStart
a-q.p	findQuickMark
a-u.p	blockUnmark
backSpace.p	prefixBackSpace
c-a.p	showAll
c-b.p.secondary	nullAction
c-backSpace.p	deleteLine
c-c.p.secondary	nullAction
c-d.p.secondary	nullAction
c-delete.p	prefixTruncate
c-e.p.secondary	nullAction
c-end.p	bottom
c-enter.p	openLine
c-f.p	find
c-g.p.secondary	nullAction
c-h.p.secondary	nullAction
c-home.p	top
c-i.p.secondary	nullAction
c-j.p	findLastChange
c-k.p.secondary	nullAction
c-l.p	locateLine

c-m.p.secondary	nullAction
c-n.p	findNext
c-o.p.secondary	nullAction
c-p.p	print
c-pageDown.p	pageRight
c-pageUp.p	pageLeft
c-q.p.secondary	nullAction
c-r.p.secondary	nullAction
c-s-a.p.secondary	nullAction
c-s-b.p.secondary	nullAction
c-s-c.p.secondary	nullAction
c-s-d.p.secondary	nullAction
c-s-e.p.secondary	nullAction
c-s-f.p	findAndReplace
c-s-g.p.secondary	nullAction
c-s-h.p.secondary	nullAction
c-s-i.p.secondary	nullAction
c-s-j.p.secondary	nullAction
c-s-k.p.secondary	nullAction
c-s-l.p.secondary	nullAction
c-s-m.p.secondary	nullAction
c-s-n.p	compareNext
c-s-o.p.secondary	nullAction
c-s-p.p	comparePrevious
c-s-q.p.secondary	nullAction
c-s-r.p	compareRefresh
c-s-s.p.secondary	nullAction
c-s-t.p.secondary	nullAction
c-s-u.p.secondary	nullAction
c-s-v.p.secondary	nullAction
c-s-w.p.secondary	nullAction
c-s-x.p.secondary	nullAction
c-s-y.p.secondary	nullAction
c-s-z.p	redo
c-s.p	save
c-t.p.secondary	nullAction
c-u.p	findUp
c-v.p.secondary	nullAction
c-w.p.secondary	nullAction
c-x.p	nullAction
c-y.p	duplicateLine
c-z.p	undo
delete.p	prefixDelete
down.p	down
end.p	prefixEnd
escape.p	commandLine
f1.p	help
f7.p	blockShiftLeft
f8.p	blockShiftRight
home.p.secondary	prefixHome
insert.p	toggleInsert
left.p	prefixLeft
pageDown.p	pageDown
pageUp.p	pageUp
right.p	prefixRight

s-tab.p	prefixHome
tab.p	home
up.p	up
a-d.c	blockDelete
a-i.c	blockLowerCase
a-k.c	blockUpperCase
a-pageDown.c	findBlockEnd
a-pageUp.c	findBlockStart
a-q.c	findQuickMark
a-u.c	blockUnmark
c-a.c	showAll
c-backSpace.c	deleteLine
c-end.c	bottom
c-enter.c	openLine
c-f.c	find
c-home.c	top
c-j.c	findLastChange
c-l.c	locateLine
c-n.c	findNext
c-p.c	print
c-pageDown.c	pageRight
c-pageUp.c	pageLeft
c-s.c	save
c-u.c	findUp
f1.c	help
f7.c	blockShiftLeft
f8.c	blockShiftRight
pageDown.c	pageDown
pageUp.c	pageUp

Mouse Event Settings

The mouse event settings listed below are divided into mouse event and action pairs. The first string (e.g. "1-a-c-pressed.1.t.p.e") indicates the mouse event and the second string indicates the action (e.g. "blockUnmark"). For a complete definition of how a mouse event is defined refer to the **mouseAction** parameter.

1-a-c-pressed.1.t	blockUnmark
1-a-c-s-pressed.1.t	blockUnmark
1-a-dragged.t	blockMarkToMouse
1-a-pressed.1.t	cursorToMouse
1-a-pressed.2.t	blockMarkRectangleAtMouse
1-c-dragged.t	blockMarkToMouse
1-c-pressed.1.t	cursorToMouse
1-c-pressed.2.t	blockMarkElementAtMouse
1-dragged.t	blockMarkToMouse
1-pressed.1.t	cursorToMouse
1-pressed.2.t	blockMarkWordAtMouse
1-s-dragged.t	blockMarkToMouse
1-s-pressed.1.t	blockMarkToMouse
2-dragged.t	blockMarkToMouse
c-dragged.t	blockMarkToMouse
dragged.t	blockMarkToMouse
popup.t	popupAtMouse
s-dragged.t	blockMarkToMouse

1-a-c-pressed.1.p
 1-a-c-s-pressed.1.p
 1-a-dragged.p
 1-a-pressed.1.p
 1-a-pressed.2.p
 1-c-dragged.p
 1-c-pressed.1.p
 1-c-pressed.2.p
 1-dragged.p
 1-pressed.1.p
 1-pressed.2.p
 1-s-dragged.p
 1-s-pressed.1.p
 2-dragged.p
 c-dragged.p
 dragged.p
 popup.p
 s-dragged.p
 1-a-c-pressed.1.e
 1-a-c-s-pressed.1.e
 1-a-dragged.e
 1-a-pressed.1.e
 1-a-pressed.2.e
 1-c-dragged.e
 1-c-pressed.1.e
 1-c-pressed.2.e
 1-dragged.e
 1-pressed.1.e
 1-pressed.2.e
 1-pressed.3.e
 1-s-dragged.e
 1-s-pressed.1.e
 2-dragged.e
 c-dragged.e
 dragged.e
 popup.e
 s-dragged.e

blockUnmark
 blockUnmark
 blockMarkToMouse
 cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkWordAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse
 blockUnmark
 blockUnmark
 blockMarkToMouse
 cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 expandHideAtMouse
 expandHideAtMouse
 expandHideAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“keyAction parameter” on page 133 parameter

“mouseAction parameter” on page 151 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

seu base profile

The **seu** base profile has key assignments and prefix commands that should be familiar to seu users. Refer to the **updateProfile.baseProfile** parameter for other base profiles.

Key Settings

The key settings listed below are divided into key and action pairs. The first string (e.g. "a-b.t") indicates the key and the second string indicates the action (e.g. "blockMarkCharacter"). For a complete definition of how a key is defined refer to the **keyAction** parameter.

a-b.t	blockMarkCharacter
a-backSpace.t.secondary	undo
a-c.t	blockCopy
a-d.t	blockDelete
a-i.t	blockLowerCase
a-j.t	join
a-k.t	blockUpperCase
a-l.t	blockMarkElement
a-m.t	blockMove
a-pageDown.t	findBlockEnd
a-pageUp.t	findBlockStart
a-q.t	findQuickMark
a-r.t	blockMarkRectangle
a-s.t	split
a-u.t	blockUnmark
a-z.t	blockOverlay
backSpace.t	backSpace (page 222)
c-a.t	showAll
c-b.t.secondary	nullAction
c-backSpace.t	deleteLine
c-c.t	copy
c-d.t.secondary	nullAction
c-delete.t	truncate
c-e.t.secondary	nullAction
c-end.t	bottom
c-enter.t	openLine
c-f.t.secondary	find
c-g.t.secondary	nullAction
c-h.t.secondary	nullAction
c-home.t	top
c-i.t.secondary	nullAction
c-insert.t.secondary	copy
c-j.t	findLastChange
c-k.t.secondary	nullAction
c-l.t	locateLine
c-left.t	prevWord
c-m.t.secondary	nullAction
c-n.t.secondary	findNext
c-o.t.secondary	nullAction
c-p.t	print
c-pageDown.t.secondary	pageRight
c-pageUp.t.secondary	pageLeft

c-q.t	setQuickMark
c-r.t.secondary	nullAction
c-right.t	nextWord
c-s-a.t.secondary	nullAction
c-s-b.t.secondary	nullAction
c-s-c.t.secondary	nullAction
c-s-d.t.secondary	nullAction
c-s-e.t.secondary	nullAction
c-s-end.t	blockMarkBottom
c-s-f.t.secondary	findAndReplace
c-s-g.t.secondary	nullAction
c-s-h.t.secondary	nullAction
c-s-home.t	blockMarkTop
c-s-i.t.secondary	nullAction
c-s-j.t.secondary	nullAction
c-s-k.t.secondary	nullAction
c-s-l.t.secondary	nullAction
c-s-left.t	blockMarkPrevWord
c-s-m.t.secondary	nullAction
c-s-n.t	compareNext
c-s-o.t.secondary	nullAction
c-s-p.t	comparePrevious
c-s-pageDown.t	blockMarkPageRight
c-s-pageUp.t	blockMarkPageLeft
c-s-q.t.secondary	nullAction
c-s-r.t	compareRefresh
c-s-right.t	blockMarkNextWord
c-s-s.t.secondary	nullAction
c-s-t.t.secondary	nullAction
c-s-u.t.secondary	nullAction
c-s-v.t.secondary	nullAction
c-s-w.t.secondary	nullAction
c-s-x.t.secondary	nullAction
c-s-y.t.secondary	nullAction
c-s-z.t	redo
c-s.t	save
c-t.t.secondary	nullAction
c-u.t	findUp
c-v.t	paste
c-w.t	nullAction
c-x.t	cut
c-y.t	duplicateLine
c-z.t	undo
delete.t	delete
down.t	down
end.t	end
enter.t	splitLine
escape.t	commandLine
f1.t	help
f5.t	clearPrefix
f7.t	pageUp
f8.t	pageDown
home.t	home
insert.t	toggleInsert
left.t	left

pageDown.t.secondary
 pageUp.t.secondary
 right.t
 s-delete.t.secondary
 s-down.t
 s-end.t
 s-enter.t
 s-f2.t
 s-f4.t
 s-f5.t
 s-f7.t
 s-f8.t
 s-home.t
 s-insert.t.secondary
 s-left.t
 s-pageDown.t
 s-pageUp.t
 s-right.t
 s-tab.t
 s-up.t
 tab.t
 up.t
 a-backSpace.p.secondary
 a-d.p
 a-i.p
 a-k.p
 a-pageDown.p
 a-pageUp.p
 a-q.p
 a-u.p
 backSpace.p
 c-a.p
 c-b.p.secondary
 c-backSpace.p
 c-c.p.secondary
 c-d.p.secondary
 c-delete.p
 c-e.p.secondary
 c-end.p
 c-enter.p
 c-f.p.secondary
 c-g.p.secondary
 c-h.p.secondary
 c-home.p
 c-i.p.secondary
 c-j.p
 c-k.p.secondary
 c-l.p
 c-m.p.secondary
 c-n.p.secondary
 c-o.p.secondary
 c-p.p
 c-pageDown.p.secondary
 c-pageUp.p.secondary
 c-q.p.secondary

pageDown
 pageUp
 right
 cut
 blockMarkDown
 blockMarkEnd
 newLine
 find
 findNext
 findAndReplace
 pageLeft
 pageRight
 blockMarkHome
 paste
 blockMarkLeft
 blockMarkPageDown
 blockMarkPageUp
 blockMarkRight
 prevTabStop
 blockMarkUp
 nextTabStop
 up
 undo
 blockDelete
 blockLowerCase
 blockUpperCase
 findBlockEnd
 findBlockStart
 findQuickMark
 blockUnmark
 prefixBackSpace
 showAll
 nullAction
 deleteLine
 nullAction
 nullAction
 nullAction
 prefixTruncate
 nullAction
 bottom
 openLine
 find
 nullAction
 nullAction
 nullAction
 top
 nullAction
 findLastChange
 nullAction
 locateLine
 nullAction
 findNext
 nullAction
 print
 pageRight
 pageLeft
 nullAction

c-r.p.secondary	nullAction
c-s-a.p.secondary	nullAction
c-s-b.p.secondary	nullAction
c-s-c.p.secondary	nullAction
c-s-d.p.secondary	nullAction
c-s-e.p.secondary	nullAction
c-s-f.p.secondary	findAndReplace
c-s-g.p.secondary	nullAction
c-s-h.p.secondary	nullAction
c-s-i.p.secondary	nullAction
c-s-j.p.secondary	nullAction
c-s-k.p.secondary	nullAction
c-s-l.p.secondary	nullAction
c-s-m.p.secondary	nullAction
c-s-n.p	compareNext
c-s-o.p.secondary	nullAction
c-s-p.p	comparePrevious
c-s-q.p.secondary	nullAction
c-s-r.p	compareRefresh
c-s-s.p.secondary	nullAction
c-s-t.p.secondary	nullAction
c-s-u.p.secondary	nullAction
c-s-v.p.secondary	nullAction
c-s-w.p.secondary	nullAction
c-s-x.p.secondary	nullAction
c-s-y.p.secondary	nullAction
c-s-z.p	redo
c-s.p	save
c-t.p.secondary	nullAction
c-u.p	findUp
c-v.p.secondary	nullAction
c-w.p.secondary	nullAction
c-x.p	nullAction
c-y.p	duplicateLine
c-z.p	undo
delete.p	prefixDelete
down.p	down
end.p	prefixEnd
enter.p	processPrefix
escape.p	commandLine
f1.p	help
f5.p	clearPrefix
f7.p	pageUp
f8.p	pageDown
home.p.secondary	prefixHome
insert.p	toggleInsert
left.p	prefixLeft
pageDown.p.secondary	pageDown
pageUp.p.secondary	pageUp
right.p	prefixRight
s-f2.p	find
s-f4.p	findNext
s-f5.p	findAndReplace
s-f7.p	pageLeft
s-f8.p	pageRight

s-tab.p	prefixHome
tab.p	home
up.p	up
a-d.c	blockDelete
a-i.c	blockLowerCase
a-k.c	blockUpperCase
a-pageDown.c	findBlockEnd
a-pageUp.c	findBlockStart
a-q.c	findQuickMark
a-u.c	blockUnmark
c-a.c	showAll
c-backSpace.c	deleteLine
c-end.c	bottom
c-enter.c	openLine
c-f.c.secondary	find
c-home.c	top
c-j.c	findLastChange
c-l.c	locateLine
c-n.c.secondary	findNext
c-p.c	print
c-pageDown.c.secondary	pageRight
c-pageUp.c.secondary	pageLeft
c-s.c	save
c-u.c	findUp
f1.c	help
f5.c	clearPrefix
f7.c	pageUp
f8.c	pageDown
pageDown.c.secondary	pageDown
pageUp.c.secondary	pageUp
s-f2.c	find
s-f4.c	findNext
s-f5.c	findAndReplace
s-f7.c	pageLeft
s-f8.c	pageRight

Mouse Event Settings

The mouse event settings listed below are divided into mouse event and action pairs. The first string (e.g. "1-a-c-pressed.1.t.p.e") indicates the mouse event and the second string indicates the action (e.g. "blockUnmark"). For a complete definition of how a mouse event is defined refer to the **mouseAction** parameter.

1-a-c-pressed.1.t	blockUnmark
1-a-c-s-pressed.1.t	blockUnmark
1-a-dragged.t	blockMarkToMouse
1-a-pressed.1.t	cursorToMouse
1-a-pressed.2.t	blockMarkRectangleAtMouse
1-c-dragged.t	blockMarkToMouse
1-c-pressed.1.t	cursorToMouse
1-c-pressed.2.t	blockMarkElementAtMouse
1-dragged.t	blockMarkToMouse
1-pressed.1.t	cursorToMouse
1-pressed.2.t	blockMarkWordAtMouse
1-s-dragged.t	blockMarkToMouse

1-s-pressed.1.t
 2-dragged.t
 c-dragged.t
 dragged.t
 popup.t
 s-dragged.t
 1-a-c-pressed.1.p
 1-a-c-s-pressed.1.p
 1-a-dragged.p
 1-a-pressed.1.p
 1-a-pressed.2.p
 1-c-dragged.p
 1-c-pressed.1.p
 1-c-pressed.2.p
 1-dragged.p
 1-pressed.1.p
 1-pressed.2.p
 1-s-dragged.p
 1-s-pressed.1.p
 2-dragged.p
 c-dragged.p
 dragged.p
 popup.p
 s-dragged.p
 1-a-c-pressed.1.e
 1-a-c-s-pressed.1.e
 1-a-dragged.e
 1-a-pressed.1.e
 1-a-pressed.2.e
 1-c-dragged.e
 1-c-pressed.1.e
 1-c-pressed.2.e
 1-dragged.e
 1-pressed.1.e
 1-pressed.2.e
 1-pressed.3.e
 1-s-dragged.e
 1-s-pressed.1.e
 2-dragged.e
 c-dragged.e
 dragged.e
 popup.e
 s-dragged.e

blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse
 blockUnmark
 blockUnmark
 blockMarkToMouse
 cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkWordAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse
 blockUnmark
 blockUnmark
 blockMarkToMouse
 cursorToMouse
 blockMarkRectangleAtMouse
 blockMarkToMouse
 cursorToMouse
 blockMarkElementAtMouse
 blockMarkToMouse
 expandHideAtMouse
 expandHideAtMouse
 expandHideAtMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse

Prefix Commands

n

+

+*n*

-

Enter *n* in the prefix area to cause the editor to reposition the cursor on the line with the specified line number. *n* is a positive integer. Enter + in the prefix area to cause the editor to scroll down one line.

Enter +*n* in the prefix area to cause the editor to scroll down *n* lines. *n* is a positive integer.

Enter - in the prefix area to cause the editor to scroll up one line.

-n	Enter -n in the prefix area to cause the editor to scroll up <i>n</i> lines. <i>n</i> is a positive integer.
/	Enter / in the prefix area to make the specified line the current line.
A	Enter A in the prefix area to cause the editor to copy or move lines after the specified line. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC , CR , CRn , CCR , M , Mn , or MM .
An	Enter An in the prefix area to cause the editor to copy or move lines after the specified line. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC , CR , CRn , CCR , M , Mn , or MM . <i>n</i> copies of the source will be inserted after the specified line. <i>n</i> is a positive integer.
B	Enter B in the prefix area to cause the editor to copy or move lines before the specified line. The source lines and type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC , CR , CRn , CCR , M , Mn , or MM .
Bn	Enter Bn in the prefix area to cause the editor to copy or move lines before the specified line. The source lines and type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC , CR , CRn , CCR , M , Mn , or MM . <i>n</i> copies of the source will be inserted before the specified line. <i>n</i> is a positive integer.
C	Enter C in the prefix area to indicate that the specified line is to be the source for a copy operation. The target location for the copy is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO .
Cn	Enter Cn in the prefix area to indicate that the specified line and the following <i>n</i> - 1 visible lines are to be the source for a copy operation. <i>n</i> is a positive integer. The target location for the copy is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO .
CC	Enter CC in the prefix area of two different lines in the document. The CC commands will delimit the source for a copy operation. Only visible lines are included in the source. The target location for the copy is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO .

CCR	Enter CCR in the prefix area of two different lines in the document. The CCR commands will delimit the source for a copy operation. Only visible lines are included in the source. The target location for the copy is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO . After the copy operation has been completed, the CCR commands are not cleared from the prefix area. They are retained to allow you to specify another target.
CR	Enter CR in the prefix area to indicate that the specified line is to be the source for a copy operation. The target location for the copy is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO . After the copy operation has been completed, the CR command is not cleared from the prefix area. It is retained to allow you to specify another target.
CRn	Enter CRn in the prefix area to indicate that the specified line and the following $n - 1$ visible lines are to be the source for a copy operation. n is a positive integer. The target location for the copy is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO . After the copy operation has been completed, the CRn command is not cleared from the prefix area. It is retained to allow you to specify another target.
D	Enter D in the prefix area to cause the editor to delete the specified line.
Dn	Enter Dn in the prefix area to cause the editor to delete the specified line and the following $n - 1$ visible lines.
DD	Enter DD in the prefix area of two different lines in the document to delimit a block of text that you want deleted. Only the visible lines will be deleted.
I	Enter I in the prefix area to cause the editor to insert a new line after the specified line.
In	Enter In in the prefix area to cause the editor to insert n new lines after the specified line. n is a positive integer.
IS	Enter IS in the prefix area to cause the editor to insert the skeleton line after the specified line. The skeleton line is specified with the S command.
ISn	Enter ISn in the prefix area to cause the editor to insert n copies of the skeleton line after the specified line. The skeleton line is specified with the S command.
L	Enter L in the prefix area to cause the editor to shift the specified line one character to the left. If there is not at least one blank at the start of the line, then nothing is done.

<i>Ln</i>	Enter <i>Ln</i> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the left. If there are less than <i>n</i> spaces at the start of the line, then only the spaces are deleted.
<i>LL</i>	Enter <i>LL</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the left. Only the visible lines will be shifted. Lines that do not have at least one space at the start of the line will be unaffected.
<i>LLn</i>	Enter <i>LLn</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the left. Only the visible lines will be shifted. Lines that have less than <i>n</i> spaces at the start of the line will only have those spaces deleted.
<i>LLT</i>	Enter <i>LLT</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the left. Only the visible lines will be shifted.
<i>LLTn</i>	Enter <i>LLTn</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the left. Only the visible lines will be shifted.
<i>LP</i>	Enter <i>LP</i> in the prefix area to cause the editor to print the specified line.
<i>LPn</i>	Enter <i>LPn</i> in the prefix area to cause the editor to print the specified line and the following <i>n</i> - 1 visible lines.
<i>LPP</i>	Enter <i>LPP</i> in the prefix area of two different lines in the document to delimit a block of text that you want to print. Only the visible lines will be printed.
<i>LT</i>	Enter <i>LT</i> in the prefix area to cause the editor to shift the specified line one character to the left.
<i>LTn</i>	Enter <i>LTn</i> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the left.
<i>M</i>	Enter <i>M</i> in the prefix area to indicate that the specified line is to be the source for a move operation. The target location for the move is determined by the first occurrence of one of the following: <i>A</i> , <i>An</i> , <i>B</i> , <i>Bn</i> , <i>O</i> , <i>On</i> , <i>OO</i> or <i>OO</i> .
<i>Mn</i>	Enter <i>Mn</i> in the prefix area to indicate that the specified line and the following <i>n</i> - 1 visible lines are to be the source for a move operation. <i>n</i> is a positive integer. The target location for the move is determined by the first occurrence of one of the following: <i>A</i> , <i>An</i> , <i>B</i> , <i>Bn</i> , <i>O</i> , <i>On</i> , <i>OO</i> or <i>OO</i> .

MM	Enter MM in the prefix area of two different lines in the document. The MM commands will delimit the source for a move operation. Only visible lines are included in the source. The target location for the move is determined by the first occurrence of one of the following: A , An , B , Bn , O , On , OO or OO .
O	Enter O in the prefix area to cause the editor to overlay the specified line. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC,CR , CRn , CCR , M , Mn , or MM .
On	Enter On in the prefix area to cause the editor to overlay the specified line and the following $n - 1$ visible lines. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC,CR , CRn , CCR , M , Mn , or MM .
OO	Enter OO in the prefix area of two different lines in the document. The OO commands will delimit the overlay target for a move or copy operation. Only the visible lines will be overlaid. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , Cn , CC,CR , CRn , CCR , M , Mn , or MM .
R	Enter R in the prefix area to cause the editor to shift the specified line one character to the right. If the line cannot be shifted without truncating a non blank character, then nothing is done.
Rn	Enter Rn in the prefix area to cause the editor to shift the specified line n characters to the right. If the line cannot be shifted n characters without truncating a non blank character, then the line will only be shifted as far to the right as it can be without truncating a non blank character.
RP	Enter RP in the prefix area to cause the editor to duplicate the specified line.
RPn	Enter RPn in the prefix area to cause the editor to duplicate the specified line n times.
RPP	Enter RPP in the prefix area of two different lines in the document. The RPP commands will delimit a block of text that you want duplicated. Only the visible lines will be duplicated.
RPPn	Enter RPPn in the prefix area of two different lines in the document. The RPPn commands will delimit a block of text that you want duplicated n times. Only the visible lines will be duplicated.

RR	Enter RR in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the right. Only the visible lines will be shifted. Lines that cannot be shifted without truncating a non blank character will be unaffected.
RRn	Enter RRn in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the right. Only the visible lines will be shifted. Lines that cannot be shifted without truncating a non blank character will only be shifted as far to the right as they can be without truncating a non blank character.
RRT	Enter RRT in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the right. Only the visible lines will be shifted.
RRTn	Enter RRn in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the right. Only the visible lines will be shifted.
RT	Enter RT in the prefix area to cause the editor to shift the specified line one character to the right.
RTn	Enter RTn in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the right.
S	Enter S in the prefix area to cause the specified line to be saved as the skeleton line. The skeleton line is used by the IS and ISn commands.
SF	Enter SF in the prefix area of an excluded block header to show the first line in the excluded block. Lines can be excluded with the X , Xn and XX commands.
SFn	Enter SFn in the prefix area of an excluded block header to show the first <i>n</i> lines in the excluded block. Lines can be excluded with the X , Xn and XX commands.
SL	Enter SL in the prefix area of an excluded block header to show the last line in the excluded block. Lines can be excluded with the X , Xn and XX commands.
SLn	Enter SLn in the prefix area of an excluded block header to show the last <i>n</i> lines in the excluded block. Lines can be excluded with the X , Xn and XX commands.
W	Enter W in the prefix area to cause the editor to scroll the text area so that column 1 is visible.
Wn	Enter Wn in the prefix area to cause the editor to scroll the text area so that column <i>n</i> is the first visible column on the left.

X	Enter X in the prefix area to cause the editor to exclude the specified line. The line can be reshown with the SF , SF<i>n</i> , SL , or SL<i>n</i> commands.
X <i>n</i>	Enter X<i>n</i> in the prefix area to cause the editor to exclude the specified line and the following <i>n</i> - 1 visible lines.. The lines can be reshown with the SF , SF<i>n</i> , SL , or SL<i>n</i> commands.
XX	Enter XX in the prefix area of two different lines in the document to delimit a block of text that you want to exclude. The lines can be reshown with the SF , SF<i>n</i> , SL , or SL<i>n</i> commands.

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“keyAction parameter” on page 133 parameter

“mouseAction parameter” on page 151 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

xedit base profile

The **xedit** base profile has key assignments and prefix commands that should be familiar to xedit users. Refer to the **updateProfile.baseProfile** parameter for other base profiles.

Key Settings

The key settings listed below are divided into key and action pairs. The first string (e.g. "a-b.t") indicates the key and the second string indicates the action (e.g. "blockMarkCharacter"). For a complete definition of how a key is defined refer to the **keyAction** parameter.

a-b.t	blockMarkCharacter
a-backSpace.t.secondary	undo
a-c.t	blockCopy
a-d.t	blockDelete
a-i.t	blockLowerCase
a-j.t	join
a-k.t	blockUpperCase
a-l.t	blockMarkElement
a-m.t	blockMove
a-pageDown.t	findBlockEnd
a-pageUp.t	findBlockStart
a-q.t	findQuickMark
a-r.t	blockMarkRectangle
a-s.t	split
a-u.t	blockUnmark

a-z.t	blockOverlay
backSpace.t	backSpace (page 222)
c-a.t	showAll
c-b.t.secondary	nullAction
c-backSpace.t	deleteLine
c-c.t	copy
c-d.t.secondary	nullAction
c-delete.t	truncate
c-e.t.secondary	nullAction
c-end.t	bottom
c-enter.t	openLine
c-f.t.secondary	find
c-g.t.secondary	nullAction
c-h.t.secondary	nullAction
c-home.t	top
c-i.t.secondary	nullAction
c-insert.t.secondary	copy
c-j.t	findLastChange
c-k.t.secondary	nullAction
c-l.t	locateLine
c-left.t	prevWord
c-m.t.secondary	nullAction
c-n.t.secondary	findNext
c-o.t.secondary	nullAction
c-p.t	print
c-pageDown.t.secondary	pageRight
c-pageUp.t	pageLeft
c-q.t	setQuickMark
c-r.t.secondary	nullAction
c-right.t	nextWord
c-s-a.t.secondary	nullAction
c-s-b.t.secondary	nullAction
c-s-c.t.secondary	nullAction
c-s-d.t.secondary	nullAction
c-s-e.t.secondary	nullAction
c-s-end.t	blockMarkBottom
c-s-f.t	findAndReplace
c-s-g.t.secondary	nullAction
c-s-h.t.secondary	nullAction
c-s-home.t	blockMarkTop
c-s-i.t.secondary	nullAction
c-s-j.t.secondary	nullAction
c-s-k.t.secondary	nullAction
c-s-l.t.secondary	nullAction
c-s-left.t	blockMarkPrevWord
c-s-m.t.secondary	nullAction
c-s-n.t	compareNext
c-s-o.t.secondary	nullAction
c-s-p.t	comparePrevious
c-s-pageDown.t	blockMarkPageRight
c-s-pageUp.t	blockMarkPageLeft
c-s-q.t.secondary	nullAction
c-s-r.t	compareRefresh
c-s-right.t	blockMarkNextWord
c-s-s.t.secondary	nullAction

c-s-t.t.secondary	nullAction
c-s-u.t.secondary	nullAction
c-s-v.t.secondary	nullAction
c-s-w.t.secondary	nullAction
c-s-x.t.secondary	nullAction
c-s-y.t.secondary	nullAction
c-s-z.t	redo
c-s.t	save
c-t.t.secondary	nullAction
c-u.t	findUp
c-v.t	paste
c-w.t	nullAction
c-x.t	cut
c-y.t	duplicateLine
c-z.t	undo
delete.t	delete
down.t	down
end.t	end
enter.t	splitLine
escape.t	commandLine
f1.t	help
f11.t	pageRight
f5.t	findNext
f6.t	find
f7.t	pageUp
f8.t	pageDown
home.t	home
insert.t	toggleInsert
left.t	left
pageDown.t.secondary	pageDown
pageUp.t.secondary	pageUp
right.t	right
s-delete.t.secondary	cut
s-down.t	blockMarkDown
s-end.t	blockMarkEnd
s-enter.t	newLine
s-home.t	blockMarkHome
s-insert.t.secondary	paste
s-left.t	blockMarkLeft
s-pageDown.t	blockMarkPageDown
s-pageUp.t	blockMarkPageUp
s-right.t	blockMarkRight
s-tab.t	prevTabStop
s-up.t	blockMarkUp
tab.t	nextTabStop
up.t	up
a-backSpace.p.secondary	undo
a-d.p	blockDelete
a-i.p	blockLowerCase
a-k.p	blockUpperCase
a-pageDown.p	findBlockEnd
a-pageUp.p	findBlockStart
a-q.p	findQuickMark
a-u.p	blockUnmark
backSpace.p	prefixBackSpace

c-a.p	showAll
c-b.p.secondary	nullAction
c-backSpace.p	deleteLine
c-c.p.secondary	nullAction
c-d.p.secondary	nullAction
c-delete.p	prefixTruncate
c-e.p.secondary	nullAction
c-end.p	bottom
c-enter.p	openLine
c-f.p.secondary	find
c-g.p.secondary	nullAction
c-h.p.secondary	nullAction
c-home.p	top
c-i.p.secondary	nullAction
c-j.p	findLastChange
c-k.p.secondary	nullAction
c-l.p	locateLine
c-m.p.secondary	nullAction
c-n.p.secondary	findNext
c-o.p.secondary	nullAction
c-p.p	print
c-pageDown.p.secondary	pageRight
c-pageUp.p	pageLeft
c-q.p.secondary	nullAction
c-r.p.secondary	nullAction
c-s-a.p.secondary	nullAction
c-s-b.p.secondary	nullAction
c-s-c.p.secondary	nullAction
c-s-d.p.secondary	nullAction
c-s-e.p.secondary	nullAction
c-s-f.p	findAndReplace
c-s-g.p.secondary	nullAction
c-s-h.p.secondary	nullAction
c-s-i.p.secondary	nullAction
c-s-j.p.secondary	nullAction
c-s-k.p.secondary	nullAction
c-s-l.p.secondary	nullAction
c-s-m.p.secondary	nullAction
c-s-n.p	compareNext
c-s-o.p.secondary	nullAction
c-s-p.p	comparePrevious
c-s-q.p.secondary	nullAction
c-s-r.p	compareRefresh
c-s-s.p.secondary	nullAction
c-s-t.p.secondary	nullAction
c-s-u.p.secondary	nullAction
c-s-v.p.secondary	nullAction
c-s-w.p.secondary	nullAction
c-s-x.p.secondary	nullAction
c-s-y.p.secondary	nullAction
c-s-z.p	redo
c-s.p	save
c-t.p.secondary	nullAction
c-u.p	findUp
c-v.p.secondary	nullAction

c-w.p.secondary
 c-x.p
 c-y.p
 c-z.p
 delete.p
 down.p
 end.p
 enter.p
 escape.p
 f1.p
 f11.p
 f5.p
 f6.p
 f7.p
 f8.p
 home.p.secondary
 insert.p
 left.p
 pageDown.p.secondary
 pageUp.p.secondary
 right.p
 s-tab.p
 tab.p
 up.p
 a-d.c
 a-i.c
 a-k.c
 a-pageDown.c
 a-pageUp.c
 a-q.c
 a-u.c
 c-a.c
 c-backSpace.c
 c-end.c
 c-enter.c
 c-f.c.secondary
 c-home.c
 c-j.c
 c-l.c
 c-n.c.secondary
 c-p.c
 c-pageDown.c.secondary
 c-pageUp.c
 c-s.c
 c-u.c
 f1.c
 f11.c
 f5.c
 f6.c
 f7.c
 f8.c
 pageDown.c.secondary
 pageUp.c.secondary

nullAction
 nullAction
 duplicateLine
 undo
 prefixDelete
 down
 prefixEnd
 processPrefix
 commandLine
 help
 pageRight
 findNext
 find
 pageUp
 pageDown
 prefixHome
 toggleInsert
 prefixLeft
 pageDown
 pageUp
 prefixRight
 prefixHome
 home
 up
 blockDelete
 blockLowerCase
 blockUpperCase
 findBlockEnd
 findBlockStart
 findQuickMark
 blockUnmark
 showAll
 deleteLine
 bottom
 openLine
 find
 top
 findLastChange
 locateLine
 findNext
 print
 pageRight
 pageLeft
 save
 findUp
 help
 pageRight
 findNext
 find
 pageUp
 pageDown
 pageDown
 pageUp

Mouse Event Settings

The mouse event settings listed below are divided into mouse event and action pairs. The first string (e.g. "1-a-c-pressed.1.t.p.e") indicates the mouse event and the second string indicates the action (e.g. "blockUnmark"). For a complete definition of how a mouse event is defined refer to the **mouseAction** parameter.

1-a-c-pressed.1.t	blockUnmark
1-a-c-s-pressed.1.t	blockUnmark
1-a-dragged.t	blockMarkToMouse
1-a-pressed.1.t	cursorToMouse
1-a-pressed.2.t	blockMarkRectangleAtMouse
1-c-dragged.t	blockMarkToMouse
1-c-pressed.1.t	cursorToMouse
1-c-pressed.2.t	blockMarkElementAtMouse
1-dragged.t	blockMarkToMouse
1-pressed.1.t	cursorToMouse
1-pressed.2.t	blockMarkWordAtMouse
1-s-dragged.t	blockMarkToMouse
1-s-pressed.1.t	blockMarkToMouse
2-dragged.t	blockMarkToMouse
c-dragged.t	blockMarkToMouse
dragged.t	blockMarkToMouse
popup.t	popupAtMouse
s-dragged.t	blockMarkToMouse
1-a-c-pressed.1.p	blockUnmark
1-a-c-s-pressed.1.p	blockUnmark
1-a-dragged.p	blockMarkToMouse
1-a-pressed.1.p	cursorToMouse
1-a-pressed.2.p	blockMarkRectangleAtMouse
1-c-dragged.p	blockMarkToMouse
1-c-pressed.1.p	cursorToMouse
1-c-pressed.2.p	blockMarkElementAtMouse
1-dragged.p	blockMarkToMouse
1-pressed.1.p	cursorToMouse
1-pressed.2.p	blockMarkWordAtMouse
1-s-dragged.p	blockMarkToMouse
1-s-pressed.1.p	blockMarkToMouse
2-dragged.p	blockMarkToMouse
c-dragged.p	blockMarkToMouse
dragged.p	blockMarkToMouse
popup.p	popupAtMouse
s-dragged.p	blockMarkToMouse
1-a-c-pressed.1.e	blockUnmark
1-a-c-s-pressed.1.e	blockUnmark
1-a-dragged.e	blockMarkToMouse
1-a-pressed.1.e	cursorToMouse
1-a-pressed.2.e	blockMarkRectangleAtMouse
1-c-dragged.e	blockMarkToMouse
1-c-pressed.1.e	cursorToMouse
1-c-pressed.2.e	blockMarkElementAtMouse
1-dragged.e	blockMarkToMouse
1-pressed.1.e	expandHideAtMouse
1-pressed.2.e	expandHideAtMouse
1-pressed.3.e	expandHideAtMouse
1-s-dragged.e	blockMarkToMouse

1-s-pressed.1.e
 2-dragged.e
 c-dragged.e
 dragged.e
 popup.e
 s-dragged.e

blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 blockMarkToMouse
 popupAtMouse
 blockMarkToMouse

Prefix Commands

"	Enter " in the prefix area to cause the editor to duplicate the specified line.
" <i>n</i>	Enter " <i>n</i> in the prefix area to cause the editor to duplicate the specified line <i>n</i> times.
""	Enter "" in the prefix area of two different lines in the document. The "" commands will delimit a block of text that you want duplicated. Only the visible lines will be duplicated.
"" <i>n</i>	Enter "" <i>n</i> in the prefix area of two different lines in the document. The "" n commands will delimit a block of text that you want duplicated <i>n</i> times. Only the visible lines will be duplicated.
(Enter (in the prefix area to cause the editor to shift the specified line one character to the left.
(<i>n</i>	Enter (<i>n</i> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the left.
((Enter ((in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the left. Only the visible lines will be shifted.
((<i>n</i>	Enter ((<i>n</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the left. Only the visible lines will be shifted.
)	Enter) in the prefix area to cause the editor to shift the specified line one character to the right.
) <i>n</i>	Enter) <i>n</i> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the right.
))	Enter)) in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the right. Only the visible lines will be shifted.
)) <i>n</i>	Enter)) <i>n</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the right. Only the visible lines will be shifted.
/	Enter / in the prefix area to make the specified line the current line.

<	Enter < in the prefix area to cause the editor to shift the specified line one character to the left. If there is not at least one blank at the start of the line, then nothing is done.
< <i>n</i>	Enter < <i>n</i> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the left. If there is are less than <i>n</i> spaces at the start of the line, then only the spaces are deleted.
<<	Enter << in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the left. Only the visible lines will be shifted. Lines that do not have at least one space at the start of the line will be unaffected.
<< <i>n</i>	Enter >> <i>n</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the left. Only the visible lines will be shifted. Lines that have less than <i>n</i> spaces at the start of the line will only have those spaces deleted.
>	Enter > in the prefix area to cause the editor to shift the specified line one character to the right. If the line cannot be shifted without truncating a non blank character, then nothing is done.
> <i>n</i>	Enter > <i>n</i> in the prefix area to cause the editor to shift the specified line <i>n</i> characters to the right. If the line cannot be shifted <i>n</i> characters without truncating a non blank character, then the line will only be shifted as far to the right as it can be without truncating a non blank character.
>>	Enter >> in the prefix area of two different lines in the document to delimit a block of text that you want to shift one character to the right. Only the visible lines will be shifted. Lines that cannot be shifted without truncating a non blank character will be unaffected.
>> <i>n</i>	Enter >> <i>n</i> in the prefix area of two different lines in the document to delimit a block of text that you want to shift <i>n</i> characters to the right. Only the visible lines will be shifted. Lines that cannot be shifted without truncating a non blank character will only be shifted as far to the right as they can be without truncating a non blank character.
A	Enter A in the prefix area to cause the editor to insert a new line after the specified line.
A <i>n</i>	Enter A <i>n</i> in the prefix area to cause the editor to insert <i>n</i> new lines after the specified line. <i>n</i> is a positive integer.

C	Enter C in the prefix area to indicate that the specified line is to be the source for a copy operation. The target location for the copy is determined by the first occurrence of one of an F or P command.
CC	Enter CC in the prefix area of two different lines in the document. The CC commands will delimit the source for a copy operation. Only visible lines are included in the source. The target location for the copy is determined by the first occurrence of one of an F or P command.
D	Enter D in the prefix area to cause the editor to delete the specified line.
Dn	Enter Dn in the prefix area to cause the editor to delete the specified line and the following $n - 1$ visible lines.
DD	Enter DD in the prefix area of two different lines in the document to delimit a block of text that you want deleted. Only the visible lines will be deleted.
F	Enter F in the prefix area to cause the editor to copy or move lines after the specified line. The source lines and the type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , CC , M , or MM .
I	Enter I in the prefix area to cause the editor to insert a new line after the specified line.
In	Enter In in the prefix area to cause the editor to insert n new lines after the specified line. n is a positive integer.
M	Enter M in the prefix area to indicate that the specified line is to be the source for a move operation. The target location for the move is determined by the first occurrence of one of an F or P command.
MM	Enter MM in the prefix area of two different lines in the document. The MM commands will delimit the source for a move operation. Only visible lines are included in the source. The target location for the move is determined by the first occurrence of one of an F or P command.
P	Enter P in the prefix area to cause the editor to copy or move lines before the specified line. The source lines and type of operation (move or copy) are determined by the first occurrence of one of the following commands: C , CC , M , or MM .
S	Enter S in the prefix area of an excluded block header to show all the lines in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.
Sn	Enter Sn in the prefix area of an excluded block header to show the first n lines in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.

S*	Enter S* in the prefix area of an excluded block header to show all the lines in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.
S+	Enter S+ in the prefix area of an excluded block header to show the first line in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.
S+n	Enter S+n in the prefix area of an excluded block header to show the first <i>n</i> lines in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.
S-	Enter S- in the prefix area of an excluded block header to show the last line in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.
S-n	Enter S-n in the prefix area of an excluded block header to show the last <i>n</i> lines in the excluded block. Lines can be excluded with the X , Xn , X* , and XX commands.
X	Enter X in the prefix area to cause the editor to exclude the specified line. The line can be reshown with the S , Sn , S* , S+ , S+n , S- , or S-n commands.
Xn	Enter Xn in the prefix area to cause the editor to exclude the specified line and the following <i>n</i> - 1 visible lines. The line can be reshown with the S , Sn , S* , S+ , S+n , S- , or S-n commands.
X*	Enter X* in the prefix area to cause the editor to exclude all of the lines in the document. The line can be reshown with the S , Sn , S* , S+ , S+n , S- , or S-n commands.
XX	Enter XX in the prefix area of two different lines in the document to delimit a block of text that you want to exclude. The line can be reshown with the S , Sn , S* , S+ , S+n , S- , or S-n commands.

RELATED CONCEPTS

“Editor commands and parameters” on page 1

RELATED REFERENCES

“Default editor commands” on page 25

“processPrefix command” on page 46 command

“Editor parameters” on page 58

“keyAction parameter” on page 133 parameter

“mouseAction parameter” on page 151 parameter

“updateProfile.baseProfile parameter” on page 194 parameter

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
Intellectual Property and Licensing
IBM Corporation
North Castle Drive, MD-NC 119
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the document. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd.
1150 Eglinton Ave E
Toronto, Ontario, M3C 1H7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and service marks

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States, other countries, or both:

CICS	IMS
CICS/ESA	Language Environment
CICS/MVS	MVS/ESA
CICS/VSE	OS/390
DB2	S/390
IBM	

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and/or other countries licensed exclusively through X/Open Company Limited.

Sun, SunLink, Solaris, SunOS, Java, all Java-based trademarks and logos, NFS, and Sun Microsystems are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

WHILE THE INFORMATION IN THIS PUBLICATION IS BELIEVED TO BE ACCURATE, THE OBJECT MANAGEMENT GROUP, AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND WITH REGARDS TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Other company, product, and service names may be trademarks or service marks of others.