# The Tioga Editor



**Abstract:** Tioga is a system to help you prepare documents. Its two main components are an editor and a typesetter. The editor lets you prepare the textual content of a document. The typesetter composes the document into pages for printing. Tioga is capable of dealing with simple technical papers and memos, and is well integrated within Cedar to support more mundane tasks such as writing programs. In future versions, it will be suitable for complex technical documents and books and will support tables, math formulas, and illustrations containing synthetic graphics and scanned images.

When viewing this document on-line, use the level-clipping functions to see the overall structure rather than simply plowing straight through. LEFT-click the "Levels" button in the top menu, then LEFT-click "FirstLevelOnly" in the new menu that appears. This will show you the major section headings. LEFT-click "MoreLevels" to see the subsections, or "AllLevels" to read the details.

**Created by:** Bill Paxton

**Maintained by:** TiogaImplementors↑.pa

**Keywords:** abbreviations, artwork, color, composition, conversion, definition search, Cedar interface, Cedar language, document model, documentation, editor, EditTool, formatting, page layout, printing, Tioga documents, TIP tables, typesetting, styles, user profile, version map, WYSIWYG

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

For Internal Xerox Use Only

# Overview

Some editors represent a document as a list of paragraphs. In Tioga, a document is a tree structure rather than a list so that you can explicitly represent its hierarchical structure. In discussing the document tree we use Computer Science terminology in which a branch is recursively defined to be a node having zero or more children branches. The root node of the document tree is not displayed – although it can be modified by a few special commands – so the document basically appears to be a list of top-level branches.

Each node in the tree contains text. The characters of the text can have *looks* which control various aspects of their appearance such as font and size. Appearance is also influenced by the *format* of the node which determines things such as vertical and horizontal spacing. The document refers to the names of looks and formats, but not to any specific interpretation of them. The interpretations are instead collected in a *style* which can be shared by many documents. For example, in the style for this document there are definitions of formats for titles, headings, and standard paragraphs, and there are definitions of looks for emphasis and for small caps. Rather than specifying all the details for the formats and looks, the document refers to them by name so it is easy to change the definitions in the style and modify the appearance uniformly throughout the document.

## User Categories

The Tioga user-interface is 'layered' so that beginning users can protect themselves from the confusion that results from mistakenly giving a command. In ways described below, you can tell the system that you are either a beginner, an intermediate, or an advanced user. In the current Cedar environment, the default is advanced, so beware.

As far as the Tioga user-interface is concerned, the user category determines which keyboard commands are currently enabled. As a beginner, you get the commands that use the special keys at the left and right of the keyboard, plus CTRL-A and CTRL-W for backspace character and word, respectively. As an intermediate user, you add a large number of commands that use print keys in combination with the various shift keys. As an advanced user, you add the keyboard commands for manipulating the document tree structure. Any category of user can get at any of the commands by using the EditTool. The user category mechanism is meant to let you protect yourself, not to limit you. The default user category in the user profile distributed with Cedar is Advanced. You are free to change your own category at any time you feel like it. For example, to declare yourself to be an intermediate user, edit your user profile file to say UserCategory: Intermediate.

## Input Devices

### *Mouse*

The mouse has three buttons named LEFT, MIDDLE, and RIGHT corresponding to their physical layout.

### *Keyboard*

The keyboard is used for commands as well as text input. Here are the names for the special keys:

LOOK top right blank key.

| | |
|---|---|
| NEXT | middle right blank key. |
| REPEAT | key labelled ESC. |
| DELETE | key labelled DEL. |
| LOAD FILE | key labelled LF. |

The bottom right blank key is an alternative CTRL key. You can use either CTRL key and either SHIFT key interchangeably.

## Scrolling and Thumbing

To move a viewer to look at a different part of the document, move the cursor into the left margin until it becomes a double arrow pointing both up and down. The part of the margin that becomes gray is called the *scroll bar*. The dark gray part shows the relative size and location of the currently visible portion of the document. LEFT-click to move the text adjacent to the arrow to the top of the viewer. RIGHT-click to move the text from the top of the viewer down to the arrow. These operations are called *scrolling* the document.

If you hold down MIDDLE in the scroll bar, the cursor becomes a right-pointing arrow. If you move the cursor to *x*% down from the top of the viewer and let up, the viewer will change to start about *x*% of the way through the document. This operation is called *thumbing*.

### *Changing levels when you scroll*

In a document such as this one that uses levels to reflect its logical structure, it is often useful to start with a view of the first level only and then progressively scroll the document and show more levels as you zoom in on a particular topic. This process is slowed when you have to move the mouse between the scroll bar and the levels menu, so we have made it possible to scroll and change levels in a single operation. This works by using the SHIFT and CTRL keys as modifiers of the LEFT-click that causes scrolling up:

Scroll up and show more levels – SHIFT-LEFT-click (hold the SHIFT key (either one) down when LEFT-clicking to scroll)

Scroll up and show all levels – CTRL-LEFT-click (hold CTRL down when LEFT-clicking to scroll)

Scroll up and show minimum levels – SHIFT-CTRL-LEFT-click (hold both the SHIFT and CTRL keys down when LEFT-clicking to scroll).

This only works if you haven't started making a secondary selection: in the case that you are, it's assumed that you are holding down the keys for another reason.

## Selections

The details of making selections will be covered later on. For now, you simply need to know that there is a single primary selection on the screen. The viewer containing the selection is referred to as the *selected viewer*. Many of the following commands deal with the selection or the selected viewer.

### Undo

While undo is described later in more detail, hitting SHIFT-ESC undoes the most recent edit sequence and restores the selection to its prior state.

### Naming Files

Tioga traffics in full FNames (see the *Naming Files* section of FSDoc.tioga). When a file has been loaded into a Tioga viewer, the viewer's caption displays the file's full FName. A Tioga viewer without a backing file displays just a working directory, ending in '>'. In either case, the working directory associated with a Tioga viewer is the part of the viewer's name up to the last '>'. For most Tioga operations that deal with files, you can specify just a short name, letting the viewer supply the working directory.

Typically you will want to view the most recent version of a file. If you omit the version part when specifying a file name, Tioga will find the most recent (!H) version. The viewer's caption will display the actual version number in parentheses. Furthermore, if another version of that file is created, Tioga will automatically reload the viewer with the newer version. (Exception: if a newer version is created while you are editing the older version, Tioga will flash a warning instead of reloading the viewer.) If you name a file with an explicit version number, Tioga will display the version number as part of the file name, and will never automatically reload the viewer.

### Emergency Save All Edits

If your machine hangs up, and you have some unsaved documents, hold down SHIFT-SHIFT-SWAT (i.e., both shift keys and the bottommost blank key) for about a second. This will save all of your edits, if possible. The cursor changes to a square while the save is taking place. Documents in unnamed viewers will be stored with names like "[]<>SaveAllEdits-*.tioga".

## Menus

### Tioga System buttons

*Open* – The Open button (located near the top right corner of the screen) performs the same function as the Get menu button described below (left column, empty working directory).

*New* – The New button creates a new empty viewer in the left column.

### Top level menu

*Clear* – creates an empty viewer.

If you LEFT-click the Clear button, the 'clicked' viewer is cleared. If you MIDDLE-click the Clear button, a new empty viewer is created below the clicked one. Finally, if you RIGHT-click the Clear button, the clicked viewer is closed and a new empty viewer appears in its place.

In the viewer's caption (at the top, where the file name would normally appear), an empty viewer will display just a working directory, ending in '>'. An empty viewer produced by Clear inherits its working directory from the 'clicked' viewer.

Naturally the most common thing to do with an empty viewer is to load a file. If you type a file name into an empty viewer and then hit LF, it is as if you selected the name and hit Get but no confirmation is required. CTRL-LF provides a similar function for GetImpl. You can also type a working directory name (ending in '>' or '/') and hit LF; this will simply change the working directory for the empty viewer.

*Reset* – discards edits by reloading the filed version of the document.

Reset is a guarded menu command to prevent accidental loss of edits. The line through the menu item represents the guard. The first click removes the guard for a short time interval and posts a confirmation message in the window at the top of the screen. A second click *while the guard has been removed* executes the command.

*Get* – loads the file named by the selection into a viewer.

If you LEFT-click the Get button, the file is loaded into the 'clicked' viewer and replaces the previous contents. If you MIDDLE-click the Get button, a new viewer is created below the clicked one in the same viewer column. Finally, if you RIGHT-click the Get button, the clicked viewer is closed to make it iconic and a new viewer appears in its place.

If the selected name is a partial name (not beginning with '[' or '/'), Tioga interprets the name in the 'clicked' viewer's working directory.

If the selected name does not include an explicit extension, Tioga will search for the file using a set of standard extensions. The default extensions are mesa, tioga, df, cm, config, and style. You can specify your own list of extensions with a user profile entry for SourceFileExtensions.

If the selected name contains a vertical bar ("|") followed by a position number, the feedback selection is set at the indicated position and document is scrolled to make it visible.

If the selected name is of the form <alpha>.<beta> and such a file exists, it is opened. Otherwise, if <beta> is one of the standard set of extensions, you will be informed that the file doesn't exist. However, if <beta> is not a standard extension, the system tries to open the file as if you had simply selected <alpha>. If this succeeds, it searches in the file for a definition of <beta> (see Def in the Places menu). This convention is intended for use with programs that have many instances of <Interface>.<Item> in which <Interface>.mesa is a file containing a definition for <Item>.

*GetImpl* – like *Get* but loads the file that implements the selected interface name.

If the selected name does not include an explicit extension (i.e., there is no period in the selected text), Tioga will use a set of standard implementation extensions. Currently, mesa is the only default implementation extension. You can specify your own list of implementation extensions with a user profile entry for ImplFileExtensions.

If the selected name is of the form <Interface>.<Item> and an implementation for the

item is currently loaded, the system will find the name of the file holding the implementation (our thanks to the Cedar runtime model for providing this information). Otherwise, the system tries to open the file as if you had selected <Interface>Impl, and, if this succeeds, searches in the file for a definition of <Item>.

*PrevFile* – like *Get* but reloads the file that was previously in this viewer.

*Store* – like *Save* but writes to the file named by the current selection.

Store is also a guarded menu command like Reset.

*Save* – writes a new version of the file.

Tioga keeps 2 versions by default when it stores files, so you can retrieve the previous version if necessary. If the file has a keep property, then Tioga keeps that number of versions. Use the SetKeep command to change the keep property on a file.

*A Comment Regarding 'Unsaved' Documents*

It is not uncommon to forget to save the contents of a viewer before destroying it or loading something else into it. However this is not a disaster since Tioga holds onto 'unsaved' documents so you can reload them with edits preserved. The number of such documents that Tioga will remember is set by a profile entry (named Tioga.UnsavedDocumentsCacheSize); the default is four.

Whenever there are unsaved documents, Tioga creates a special viewer named 'Unsaved Documents List' containing the names of the unsaved documents. To retrieve a document with all the unsaved edits, select the entire filename, including a version number if one, and click Open in the system menu area at the top right of the screen.

Unsaved documents are put into the cache by Destroy, Clear, Get, GetImpl, and PrevFile. They are *not* put in by Reset. If the cache is already full when a new entry arrives, the oldest entry is discarded. Whenever a file is needed for Get, GetImpl, or PrevFile, the cache is checked to see if an unsaved version is available. Unnamed documents are not put into the cache.

*Time* – inserts the current time at the caret.

*Split* – creates a new viewer looking at the same document.

The selection is highlighted in one viewer only, however edits will be reflected in all viewers for the document. Note that the split viewers can be independently closed, opened, or moved on the screen, and may be displayed at different levels of detail (see the Levels menu section).

*Places, Levels* – show/remove these submenus.

You may notice additional menu items after Levels: this are placed there by other programs (e.g., EditorComforts), and are documented elsewhere. Also note that other programs may alter the behavior of the normal Tioga menu items

## Places

The Places menu contains commands that cause the viewer to begin displaying at a new place in the document.

For the first three of the commands (Find, Word, and Def) commands, the button used in clicking the menu item determines how the search is carried out – LEFT-click to search towards the end of the document, RIGHT-click to search towards the start of the document, or MIDDLE-click to search first towards the end and, if that fails, then from the start of the document towards the selection. If the selection is visible in the viewer, the search starts there. Otherwise, it starts from the top of the viewer.

*Find* – find another instance of the selected text.

In this command and the following two, the direction of search and capitalization are determined by the mouse buttons and SHIFT keys. LEFT-clicking searches forward from the current selection point; RIGHT-clicking searches backwards; MIDDLE-clicking does a wrap-around search by first searching forward to the end of the document then searching from the beginning until the entire document has been searched. By default, the search matches capitalization. Clicking with the SHIFT key down invokes a 'caseless' search in which capitalization does not matter.

*Word* – find an instance of the selected text that is a 'word'.

A word is defined to not have adjacent letters or digits.

*Def* – find a 'definition' of the selected text.

A definition is an instance of the selected text that is immediately followed by a colon and isn't immediately preceded by an alphanumeric.

*Position* – scroll to and select the character position indicated by the selected number.

This is useful with compiler error messages that give locations as character counts. The command scrolls to the selected character number and then selects it – e.g., if '183' is selected, scroll to and select character number 183 in the document.

Compiler error messages give locations that exclude comment nodes. For error messages that include comment nodes, such as those from the TSetter, select the number and SHIFT-click Position to include comment nodes.

*Normalize* – scroll to make the selection visible.

LEFT-click to scroll to the start of the selection, RIGHT-click to scroll to the end, or MIDDLE-click to scroll to the caret. If the document in this viewer does not contain the selection, scrolls to the start of the document.

*PrevPlace* – scroll to place that was previously visible.

Go back to the place that was previously visible in this viewer discounting manual scrolling that may have taken place since.

*Reselect* – restore the most recent selection in this viewer and scroll to it.

*StyleKind* – control kind of style (screen or print) in the viewer.

LEFT-click to get screen style (legible on screen). RIGHT-click to get print style (fonts, spacing, line breaks more closely match printed output). Note that different style kinds may be used in different viewers on the same document.

### Levels

These commands in the Levels menu let you control how deep the display goes in the document tree structure.

*FirstLevelOnly* – show only the top level nodes.

*MoreLevels* – show one more level than currently.

*FewerLevels* – show one fewer level than currently.

*AllLevels* – show all levels of the tree.

## Selections

Tioga performs its editing operations on selected portions of the document. The normal method of editing in Tioga is to make a selection and then request an operation. The selection may be changed at any time prior to requesting the operation be completed.

Two kinds of selections are most frequently used: *primary* and *secondary*. A third kind of *feedback* selection provides feedback for tools that manipulate Tioga documents, such as the debugger identifying the point in a source file corresponding to the current execution stack position by scrolling the source file document and posting a feedback selection.

### Primary selections

The primary selection is the one that's around most of the time and is the usual site for edits. It is displayed with a solid underline or with video reverse. Make a primary selection with the mouse in ways described below.

*Insertion point*

The insertion point goes with primary selection. It is shown by a blinking 'caret' at one end or the other of the selection – the end closer to the cursor when the selection was made or the end most recently extended.

### Secondary selections

The secondary selection is a selection made with the SHIFT key held down. It is displayed with a gray underline or with a gray background. Secondary seletions are often involved in Copy and Move operations.

## Making selections

*The selection hierarchy*

The selection hierarchy consists of the following granularities: point, character, word, node, branch, and document.

*Point selection*

The primary selection has a blinking caret at one end. Some operations, such as delete and type-in, reduce the selection to just the caret. This is called a point selection.

*Character selection*

LEFT-click with the cursor over the desired character. You can hold the LEFT button down and move to the correct place before letting the button up.

You can cancel the new selection by hitting DEL while the mouse button is still down. Tioga will restore the previous selection.

*Word selection*

A 'word' is defined as a sequence of letters and digits or a sequence of identical characters that are not letters or digits. Use the MIDDLE mouse button to make word selections. As with character selection, you can hold MIDDLE down and move to the correct word before letting up.

*Node selection*

LEFT-double-click to select a node.

*Branch selection*

A branch is a node and any children branches it might have. MIDDLE-double-click to select a branch.

*Document selection*

CTRL-D extends the selection to include the entire document. 'D' stands for for 'Document'.

*Selection extension*

Extend an existing selection by pointing at a new endpoint either before or after the selection and RIGHT-clicking. The system will extend the end of the selection closer to the cursor when RIGHT goes down. You can hold RIGHT down and move the endpoint to a new position.

If you just RIGHT-click, the selection is extended at the same level in the selection hierarchy. For example, a selection at the word level will be extended a word at a time. Double-RIGHT-click to extend at a lower level in selection hierarchy. Triple-RIGHT-click

to extend at a higher level in selection hierarchy. Thus, if you have a node-level selection and wish to extend it to a word position within a node, double-RIGHT-click to reduce the level to words and then do the extension. If necessary, you can then double-RIGHT-click again to reduce to character level.

## Editing by making selections

### *Delete selections*

A *delete selection* is deleted as soon as the selection is completed. It is shown as video reverse. Hold down CTRL to make a delete selection. The selection is complete when you let up on both the mouse button and the CTRL key.

### *Pending-delete selections*

'Pending-delete' selections are automatically deleted by subsequent insertions. They are shown with video reverse rather than solid underline. Whenever you extend a selection by RIGHT-clicking, the selection is automatically made pending-delete. Notice that you don't have to actually point to a new place in the text to 'extend' a selection. For example, you can MIDDLE-click to select a word and then RIGHT-click without moving the mouse to make it pending-delete. Or you can combine these actions by 'rolling' from the LEFT or MIDDLE mouse button to the RIGHT button to make a char or word selection pending-delete.

### *Copy and Move*

Copy and Move operations require two selections, one for the source of the operation and the other for the destination. The Primary and Secondary selections are used for these. Secondary selections are made with the SHIFT key held down and are shown with a gray underline.

#### Copy secondary to primary

If you hold down the SHIFT key and select, the source selection will be copied. If the primary selection is currently pending-delete, it will be replaced by the copy of the source. Otherwise, the copy will be inserted at the caret.

#### Move secondary to primary

If you hold down both the SHIFT key and the CTRL key, the source selection will be moved. As before, if the primary selection is pending-delete, it will be replaced by the source. Otherwise, the source will be moved to the caret.

### *Destination selections for Copy and Move*

The operations described above work by copying or moving a secondary selection to the primary selection. However, sometimes the primary selection is itself the thing you want to copy or move. The following two commands take care of these situations.

#### Copy primary to secondary

To copy the primary selection, hit CTRL-S, and with the CTRL key still held down,

select a destination. The copy takes place as soon as you let the keys up. The primary selection is made not-pending-delete as soon as you hit CTRL-S to indicate that it will be copied rather than moved.

Move primary to secondary

To move the primary selection, hit CTRL-Z, and with the CTRL key still held down, select a destination. The move takes place as soon as you let the keys up. The primary selection is made pending-delete as soon as you hit CTRL-Z to indicate that it will be moved rather than copied.

*Transpose selections*

We now have covered commands to copy or move either the primary or the source selections. A final option is to transpose the primary and the source. To do this, hit CTRL-X, and with the CTRL key still held down, select a source. The primary selection is made pending-delete as soon as you hit CTRL-X to indicate that it will be moved rather than copied. The transpose takes place as soon as you let the keys and mouse buttons up.

## Miscellaneous

*Canceling a selection*

Hit DEL before finishing the selection and the previous selection will be restored. This works during either primary, source, destination, or transpose selections.

*Placeholders*

A 'placeholder' is all the text between a matching pair of placeholder brackets, ▶ ◀. Hit NEXT to find and select the next placeholder beyond the current selection. Hit SHIFT-NEXT to find the previous one. Recall that NEXT is the blank key to the right of RETURN.

If there isn't another placeholder, NEXT will move the selection to the end of the document. If the selection is already at the end, NEXT will try to find the next nested text viewer. Similarly, SHIFT-NEXT will move the selection to the start of the document if it doesn't find a previous placeholder and will try to find the previous nested text viewer if it is already at the start. This allows you to use NEXT both when filling in placeholders within a document and when filling in text viewers in tools.

*Select visible – expand selection to blanks*

CTRL-V expands the selection to include the 'visible' characters on the left and right ends.

This is useful for selecting things like full file names. For example, in the following you could make a character selection anywhere in the name and then extend the selection with CTRL-V.

Filed on: [Cedar]<Cedar7.0>Documentation>TiogaDoc.tioga

*Select matching brackets*

CTRL-] extends the selection to the left and right to find a matching pair of [..]'s. Similarly for CTRL-}, CTRL-), and CTRL->. Note that you hit CTRL with the right bracket to extend the selection. In addition, you can hit CTRL with the left bracket to insert matching brackets around the selection.

Unfortunately, our keyboards don't have both left and right quote keys. However most of the fonts, including TimesRoman and Helvetica, do provide a left and right single quote. The keyboard quote key inserts a right single quote (code 047); the left single quote (code 140) can be inserted using the MakeOctalCharacter command in the EditTool or with the Insert Matching Single Quotes command (CTRL-'). The EditTool also has a command which extends the selection to find a matching pair of left and right single quotes. The situation for double quotes is even less uniform. Some fonts, such as Classic, have a left double quote (code 264) in addition to a right double quote (code 042). However, most fonts have only the 042 double quote, so the Tioga commands for inserting and matching double quotes use that code exclusively.

# Editing

## Text input

Typed-in characters are inserted at the caret.

To insert the current time use CTRL-T – 'T' for *Time*.

When you insert a carriage return by hitting RETURN, the system will automatically copy the blank characters (tabs and spaces) from the start of the previous line. To suppress this, type SHIFT-RETURN and only the carriage return will be inserted.

To insert control characters or characters with a specific octal code, use CTRL-K or CTRL-O. The former will change the character before the caret to a control character, while the latter will convert the three digits before the caret to the corresponding octal character. The inverse operations are also available as CTRL-SHIFT-K and CTRL-SHIFT-O.

The Xerox Network Systems architecture provides for 16-bit character codes as an extension of the familiar 8-bit character codes. Tioga allows such character codes with the MakeOctalCharacter (CTRL-O) and UnMakeOctalCharacter (CTRL-SHIFT-O) commands just mentioned, by using character strings of the form "(sss|ccc)" in place of the three-digit octal character. In this string, "sss" represents the three-digit octal character set and "ccc" represents the three-digit octal character code. The interpretation of 16-bit character codes according to the Xerox Character Code Standard is enabled by providing the appropriate value to the fontPrefix style parameter to select the appropriate set of fonts.

Tioga does not use the string-body encoding when saving the files, so the character set information is not seen by programs that only read the text portions of Tioga documents. Also, certain of Tioga's operation (string searches, for example) do not use the character set information. These shortfalls may be fixed someday.

## Abbreviation Expansion

CTRL-E – 'E' for *Expand abbreviation*

When you hit CTRL-E, the caret is moved to the right of the selection if necessary and the keyname to the left of the caret is then replaced by the expansion text (according to the definition which is linked to the style in a manner described below). If the keyname had looks, they are added to the expansion. If the keyname was all caps, the expansion is made

all caps too. If the keyname had an initial cap, the first character of the expansion is made uppercase (useful at the start of sentences, for example). If the definition node has a non-null format, the format of the caret node is changed to be the same as the definition node. If the expansion contains a placeholder, the first placeholder is selected. Otherwise the entire expansion is selected.

Definitions for abbreviations come from Tioga documents which are automatically read by the system when needed. The name of the appropriate abbreviations file is determined by the style that is in effect at the caret when the expansion takes place. For example, if the style is 'Report', the abbreviations will come from the file 'Report.Abbreviations' (c.f. the section on Style Search Rules under the profile options below). You can override this by explicitly naming the abbreviations file along with the keyname. For example, 'Mesa.proc' will expand the abbreviation for 'proc' from the file 'Mesa.Abbreviations' independent of the style in effect at the caret. (Fine point: Since the system interprets a period before the keyname to mean that you're specifying a particular abbreviations file, you cannot type a vanilla abbreviation after a period.)

Each definition in an abbreviations file consists of a separate branch. The top node of the branch holds the keyname followed by an equals sign and then the text expansion. The rest of the branch, if any, is copied after the caret node as part of expanding the abbreviation; if the text expansion following the keyname is empty, then the rest of the branch will be at the same level as the caret node, otherwise, the rest of the branch is nested underneath the caret node. Any text following the keyname is moved to the end of the last child node in the branch. The definition may also include a list of operations to be performed after the expansion has been inserted. These operations have the same format as those in EditTool and are placed in parentheses after the keyname and before the equals sign.

### Delete Character or Word

BackSpace: CTRL-A, CTRL-H, or BS. Deletes the character to the left of the caret.

If the caret is at the start of a node, this does a Join command (q.v.).

BackWord: CTRL-W, or CTRL-BS. Deletes the word to the left of the caret.

DeleteNextChar: CTRL-SHIFT-A, CTRL-SHIFT-H, or SHIFT-BS. Deletes the character to the right of the caret up to the end of a node.

DeleteNextWord: CTRL-SHIFT-W, or CTRL-SHIFT-BS. Deletes the word to the right of the caret up to the end of a node.

### Delete

As mentioned above, you can delete something by selecting it with CTRL held down. In addition, you can delete the current selection by hitting DEL.

### Paste

Paste: CTRL-P. The most recently deleted text is copied to the caret. You can also use the EditTool to save the current selection to be pasted later.

### Copy, Move, Replace, and Transpose

These operations are all carried out by making selections. They are described in detail in the previous section.

### Insert matching brackets

CTRL-[ adds a matching pair of [..]'s to the ends of the selection. Similar CTRL commands exist for {, (, <, -, `, and ". CTRL-B inserts matching placeholder brackets ▶◀.

Note: CTRL-` inserts a left single quote (code 140) and a right single quote (code 047); CTRL-" inserts the same character (code 042) at each end of the selection.

### Case

All lower: CTRL-C. Makes the selection all lower case.

All caps: CTRL-SHIFT-C. Makes the selection all upper case.

First cap: CTRL-double C. Capitalizes the first word of the selection

Initial caps: CTRL-SHIFT-double C. Capitalizes each word in the selection.

### Repeat

Hitting ESC will repeat the most recent non-empty edit sequence starting with the current selection. Edit sequences are separated by user-made selections. For example, if you select a word, delete it, type a new one, and then select something else, the edit sequence is delete followed by text entry. If you hit Repeat, the system will do a delete and retype the new word.

Auto-repeat is done by ESC-select: if you hold down the ESC key while making a selection, a Repeat will automatically be done as soon as the selection is completed. This is useful when you're doing a large number of repeats.

### Undo

Hitting SHIFT-ESC undoes the most recent edit sequence and restores the selection to its prior state. If you want to undo more than just the most recent sequence, use the EditHistory tool which is described later.

### Tree structure editing

As explained in the introduction, Tioga documents consist of a tree of nodes. The following commands let you break, join, and nest nodes in the tree.

Break: CTRL-RETURN – break node at insertion point to create a new node.

Join: CTRL-J – join node at insertion point with previous node.

Nest: CTRL-N – move selected nodes to deeper nesting level in tree.

UnNest: CTRL-SHIFT-N – move selected nodes to shallower nesting level in tree.

Break & Nest: CTRL-I – simultaneously insert a new node and nest it.

Break & UnNest: CTRL-SHIFT-I – simultaneously insert and unnest.

## Looks

Characters have looks which are named by the lower case letters 'a' to 'z'. Looks are interpreted by the style to change the appearance of the text. For example, look 'e' might stand for *'emphasis'* and might result in italic face in one style and bold face in another. Each character has a set of looks – thus it may have several looks simultaneously, but each look occurs only once. You can use the EditTool to read or change the set of looks for selected characters. The following keyboard commands are also available for dealing with looks.

### Selection Looks

You can change the selection looks with the following commands. (Recall that the LOOK shift is the top blank key to the right of BS.)

LOOK-char to add to selection looks.

LOOK-SHIFT-char to remove from selection looks.

LOOK-space to remove all selection looks.

### Caret Looks

Caret looks determine the looks of typed-in text. The caret picks up the looks of the adjacent selected text whenever a selection is made. Changing the selection looks also changes the caret looks. However, if you wish to change the looks of the caret without changing the selection looks, LEFT-click the character key twice in quick succession.

LOOK-char-char to add to caret looks.

LOOK-SHIFT-char-char to remove from caret looks.

LOOK-space-space to remove all caret looks.

### Using Selections To Copy Looks

To copy the looks of some existing text to the primary selection, hit CTRL-Q, and then with the CTRL key still held down, select the text with the looks you want to copy. The source looks replace any looks the selection previously had.

### Automatic Mesa formatting

The CTRL-M command scans the selection for Mesa keywords, comments, and procedure names and gives them looks k, c, and n respectively. (As a convenience during typein, the entire caret node is reformatted if the selection is a single character or less.) With the standard Cedar style, the keywords will then be displayed in small caps, the comments will be italic, and the procedure names will be boldface. We expect to provide more extensive reformatting capabilities in the future.

## Formats

Just as characters have looks, nodes have formats. The 'format' is the name of a rule in the style that tells how to modify various parameters when displaying the node. For example, a style for documents might contain formats for titles, headings, quotations, standard paragraphs, etc. The EditTool has facilities for reading and changing the format of a node, or you can use the commands described below. (By convention, the null format name is equivalent to 'default'.)

### Setting and inserting caret node format

CTRL-← will delete the word to the left of the caret and make it the format of the caret node.

In most cases you will give this command immediately after typing the format name, so the caret will naturally be in the correct place. However, to handle cases in which you select the name before hitting CTRL-←, the caret will automatically be forced to the right of the selection at the start of this command.

CTRL-SHIFT-← inserts the format name at the caret.

This gives you a simple way to find out the format of a selected node.

### Using selections to copy formats

To copy the format of some exisiting node to the selection nodes, hit CTRL-F, and then with the CTRL key still held down, select the node with the format you want to copy.

## The EditTool

The EditTool provides a variety of operations on Tioga documents.

The text fields in the EditTool follow the convention that clicking the field name with LEFT causes the contents of the field to be selected pending-delete while clicking with RIGHT causes the field to be cleared and selected.

### Search and Substitute

*Search*

To do a search, enter the text you're looking for in the 'Target' field, select where you want the search to start, and LEFT-click 'Search' to search forward, RIGHT-click to search backwards, or MIDDLE-click to search first forward then backwards. The system searches from the current selection and updates the selected viewer if the search succeeds. (Fine point: in both searches and substitutes, the match is limited to a single node – we do not yet have mechanisms for doing matches across node boundaries.)

The multiple choices below the 'Replacement' field control what is matched in searches and replaced in substitutes. You can select or deselect an item by clicking it with the mouse. White text on black background means that the item is selected; black text on white means it is not selected.

Text – if this option is selected, match characters of target text when searching.

Looks – if selected, match looks of target text when searching.

Format – match format of target node.

Style – match style of target node.

Comment – match comment property of target node.

For example, if you pick the Looks option and deselect the Text option, you can search for any text that has a particular set of looks. If you pick only Text, the matching will ignore the looks of the target. If you pick Text and Looks, the matching text must match both the characters and the looks of the target text.

The other options deal with node properties. If you pick Format, the matching will be limited to nodes with the same format as the target node. Similarly, if you pick Style, the matching will only look at nodes whose style is the same as the target node's. Finally, if you pick Comment, the match will consider only nodes with the same value of the Comment property (TRUE or FALSE) as the target.

The first two rows of boxes below the multiple choices give you control over how matching is performed. LEFT-click with the cursor over a box to change the choice next to it. The various choices are as follows:

1. Case of matching text
   - Match Case – matching text must have same case as target text.
   - Ignore Case – matching text does not have to have same case as target text.
2. Interpretation of target text
   - Match Literally – don't treat target text as a pattern.
   - Match as Pattern – do treat target as pattern. (Patterns are described below.)
3. Context of target text
   - Match Anywhere – ignore context of matching text.
   - Match Words Only – matching text must not have adjacent letters or digits.
   - Match Entire Nodes Only – matching text must span entire node.
4. Matching target looks
   - Subset as Looks Test – looks of matching text must include target looks.
   - Equal as Looks Test – looks of matching text must be identical to target looks.

*Substitute*

To do a substitution, enter the new text in the 'Replacement' field, enter the text to be replaced in the 'Target' field, and hit 'Substitute' in the menu at the top of the EditTool. The Text/Looks/Format/... options guide the search in the usual manner and also control what is replaced.

If you pick Text and Looks, the matching text will be replaced just as if you had selected it with pending delete and made a source secondary selection of the replacement text.

If you pick only the Looks option, the matching text will have the target looks

removed and the replacement looks added.

If you pick only Text, the replacement text will have the looks of the replaced text added to it. (Fine point: if the looks of the replaced text are not uniform, the looks of the first character will be used throughout.)

If you pick Format, the matching node will get the format of the replacement node. Similarly, picking Style causes the matching node to get the style of the replacement node, and picking Comment causes the matching node to get the same value of the Comment property as the replacement node.

The final three boxes in the Search&Substitute section give you further control over this operation.

1. First character capitalization of the replacement text

   First cap like replaced – if the replaced text starts with a capital letter, force the first letter of the replacement to be a capital too.

   Don't change caps – leave the replacement capitalization alone.

2. What is done to the matching text

   Do Replace – do the usual substitute or replace.

   Do Operations – instead of doing a replace, select the matching text and then do the operations currently in the 'Operations' field of the EditTool.

3. Where the substitutions will take place

   Within Selection Only – substitute is limited to current selection.

   After Selection Only – substitute after selection to end of document.

   In Entire Document – substitute in the entire selected document.

Case-by-Case Substitutes

In addition to doing global substitutes, you can decide on a case-by-case basis whether or not to replace the matching text by the new text. Use the search commands to find the first matching text. Then if you hit 'Yes', the system will do a 'Replace' followed by a search forward. If you hit 'No', it will skip the 'Replace' and simply do another search. Thus to selectively substitute, start with a search, then do 'Yes' for the cases you want to change and 'No' for the others. The 'Replace' command simply does for the current selection what a substitute would do for a match. Finally, the 'Count' command is available to tell you how many substitutions would take place without actually changing the document.

In order to do a search or substitute, you will typically need to fill in the Target or Replacement fields in the EditTool. Naturally, this changes the selection, and before you can do the operation, you must restore the selection to the place where you actually want it to take place. The system helps you with this by saving the primary selection if it is not in the EditTool when you LEFT-click either the Target or the Replacement button. The commands along the top of the EditTool – Search, Substitute, Yes, No, Replace, and Count – restore the saved selection if the primary selection is in the EditTool when they are clicked. The net effect is that if you start out with the selection in the right place, you can LEFT-click the Target or Replacement buttons, fill in the needed information, and then directly click one of the commands without needing to reselect since the system will do it for you.

*Patterns for Search and Substitute*

When you specify that the target be matched as a pattern rather than literally, the following symbols in the target text are interpreted specially. A short summary of these symbols appears at the bottom of the Search&Substitute section of the EditTool.

Characters

- # Match any single character.
- * Match shortest possible sequence of characters.
- ** Match longest possible sequence of characters.
- ` Match the next character in the pattern exactly.
- ~ Match any character except the next one in the pattern.

Alphanumeric characters

- @ Match any single alphanumeric character (letter or digit).
- & Match shortest possible sequence of alphanumeric characters.
- && Match longest possible sequence of alphanumeric characters.

Non-alphanumeric characters

- ~@ Match any single non-alphanumeric character.
- ~& Match shortest possible sequence of non-alphanumeric characters.
- ~&~& Match longest possible sequence of non-alphanumeric characters.

Blank characters

- % Match any single blank character.
- $ Match shortest possible sequence of blank characters.
- $$ Match longest possible sequence of blank characters.

Non-blank characters

- ~% Match any single non-blank character.
- ~$ Match shortest possible sequence of non-blank characters.
- ~$~$ Match longest possible sequence of non-blank characters.

Miscellaneous

- | Match start or end of node.
- { Mark start of resulting selection.
- } Mark end of resulting selection.
- < Mark start of named subpattern.
- > Mark end of named subpattern.

The named subpatterns are of use in substitutes that reorder or duplicate parts of the matching text. The full syntax for a named subpattern is <name:subpattern>. The special case of 'match any sequence of characters' is provided as a default – i.e., <name> is equivalent to <name:*>. As far as the matching is concerned, the occurrence of <name:subpattern> is the same as if the subpattern had appeared without a name, but it has the side-effect of remembering the subsection it matched. The 'Replacement' field can contain <name>'s corresponding to named subpatterns in the target. The replacement text is constructed by replacing the <name>'s with the section of replaced text that matched the subpattern. The replacement is automatically considered to be a pattern whenever the target is – you don't need to do anything special to get the <name>'s in the replacement interpreted as subpatterns.

For example, if the target is

Target: WHILE <pred>[<arg>] DO

and the replacement is

Replacement: WHILE <arg> IS <pred> DO

then 'WHILE blue[moon] DO' will be converted to 'WHILE moon IS blue DO'.

## Looks, Formats, Styles, and Properties

### *Looks*

The Looks commands let you read and modify the looks of the caret or the selection. The looks are shown as a series of letters in the 'Looks characters' field. The box lets you pick whether you want the looks for the selection or the looks for the caret.

Get – fills the 'Looks characters' field with the letters for the caret/selection looks.

Set – reads the 'Looks characters' field and sets the looks of the caret/selection.

Clear – removes all looks from the caret/selection.

Add – adds the specified looks to the caret/selection.

Sub – removes the specified looks from the caret/selection.

### *Formats*

The Format commands let you read and modify the format for the root node or the selected nodes. The box at the right lets you pick the case you want.

Get – fills the 'Format name' field.

Set – reads the 'Format name' field and sets the node's format.

Clear – removes the node's format name. This is the same as specifying 'default' format.

### *Styles*

A style is a collection of interpretations for looks and formats. The definition of style rules are described later in their own section. The Style commands in the EditTool let you read and modify the name of the style for the root node or the selected nodes. The new style applies to the specified node and all the nodes within its sub-branches that do not themselves have explicit styles.

Get – fills the 'Style name' field.

Set – reads the 'Style name' field and sets the node's style.

Clear – removes any style specification from the node.

The following two buttons won't normally be needed: see "Style Search Rules" below.

LoadStyleDefinition – reads the 'Style name' field and reloads the style definition from that file with extension 'Style'. (Don't put the extension in the field – just put the style name and let the system add the extension.)

LoadAbbreviations – reads the 'Style name' field and reloads the abbreviation definitions from that file with extension 'Abbreviations'.

*Properties*

A node can have an arbitrary set of 'properties'. Each property consists of a name and a value. Certain properties are used by the system, but you (and your programs) are free to add others.

The 'Property name' field specifies the name of a property. (Incidentally, case distinctions do matter in property names – 'foo' is a different property than 'Foo'.) The 'Property value' field specifies a value. The box lets you pick whether you are talking about properties of the root node or the selected nodes.

Get – fills the 'Property value' field with the current value of the named property.

Set – reads the 'Property value' field and sets the property value.

Remove – removes the named property from the node.

List – fills the 'Property name' field with the names of the node's properties.

Properties may also be applied at the character level; click the box until it says "For selected characters". Character properties are not always interpreted the same way as the node properties, but the Postfix property (see below) works for both.

In addition to setting and reading properties, there is a mechanism that lets you find nodes with a certain property value. For example, if you have annotated a document with comments under the property name 'MyOpinion', and you want to find nodes in which your comment included the word 'good', enter the name in the 'Property name' field and the text in the 'Value pattern' field. Then use the 'Find' command to search forward or backwards from the caret node for a node with a value of the property that matches the pattern. (LEFT-click to search forward, RIGHT-click to search backwards.) Value patterns can use the standard set of special characters for searches. If a match is found, the node is selected and the property value is displayed.

*Reserved Properties*

Tioga and some other commonly-used programs use several property names for representing structure. Many of these have values that are simply human-readable character strings, but some are meant to be read and written only by programs, and so you should not try changing them with the EditTool unless you really understand what you are doing. Others are non-persistent (i.e. do not get saved in files) and are read and written only by programs. Note that programs other than Tioga may register new kinds of properties, so the list here is not exhaustive. The reserved properties are:

| | |
|---|---|
| Artwork | registered name of artwork class (ok to edit) |
| Bounds | used by some Artwork implementations for describing the bounding box of some artwork (edit with care) |
| CharProps | character properties (do not edit directly; see above for how to change character properties) |
| CharSets | character sets for XNS 16-bit character codes (do not edit directly) |
| Comment | 'TRUE' or 'FALSE' (ok to edit; see below) |
| DocumentLock | (non-persistent) |

| | |
|---|---|
| FileCreateDate | (non-persistent) |
| FileExtension | (non-persistent) |
| Format | format of node (ok to edit; see above) |
| Interpress | an Interpress master used by ArtworkInterpress implementation (do not edit) |
| LockedViewer | (non-persistent) |
| Postfix | (ok to edit; see below) |
| Prefix | (ok to edit; see below) |
| StyleDef | style definition (ok to edit; see below) |
| TiogaNodeAddrsProp | (non-persistent) |
| Viewer | (non-persistent) |

*Comment Property*

All nodes have a 'Comment' property which is either 'TRUE' or 'FALSE'. If its value is TRUE, the text of the node is not seen by programs such as the compiler that are only interested in the basic text contents. This makes it possible to intermix documentation and program text without requiring special escape characters to mark the start and end of comments. You can set the Comment property by using the EditTool or with the following commands.

CTRL-\ – set comment property of all selected nodes to TRUE

CTRL-SHIFT-\ – set comment property of all selected nodes to FALSE

If you are worried about not knowing at a glance what is a comment node and what is not, find a StyleRule in the style that applies to all the formats you are interested in (e.g., 'standard' in Cedar.style), and insert the line

```
isComment {visible underlining} {none underlining} .ifelse .cvx .exec
```

This will cause the contents of all comment nodes to be underlined. (You might want to say 'strikeout' instead of 'underlining' if the style already uses underlining to mean something else, and legibility is not your main concern.)

## Miscellaneous

*Sort and Reverse*

These commands let you sort or reverse lists of things. The 'Sort' command sorts things in alphabetical order, ignoring case. 'Sort-and-remove-duplicates' is useful when you are merging sets of things. The box below the Sort button lets you pick whether you want increasing or decreasing order in the result. The right box lets you pick what will be sorted: text delimited by blanks, lines delimited by carriage returns, or branches of the document tree. Leading blanks are ignored when sorting lines or branches.

*Operations*

These commands let you construct simple edit macros. The 'Operations' field holds a text description of a command sequence. The 'GetLast' command fills in the

Operations with the description of the most recent sequence, i.e., the one a Cancel would cancel or a Repeat would repeat. 'Do' executes the description in the operations field. 'Begin' marks the start of a command sequence. 'End' fills the Operations with the description of the commands since the most recent 'Begin'. 'SetCom' stores the operations under the CTRL-number key for the number in the 'Command [0..9]' field. Conversely, 'GetCom' fills the 'Operations' field with the current CTRL-number definition.

To see how this works, select the following word: 'hello'. Hit DEL and type in 'howdy'. Now hit the 'GetLast' button. The Operations field should now contain: Delete 'howdy'. Edit the Operations field contents to replace 'howdy' by 'goodbye', then hit the 'SetCom' button (first put 1 in the Command field if it's not already there). Now select the 'howdy' you typed earlier and hit CTRL-1. If all went well, the 'howdy' will have been deleted and 'goodbye' inserted in its place. More examples of edit macros will be given later.

*Searches and Operations on Files*

You can use the EditTool to look through a list of files for one in which the current search specifications are satisfied. This can be accomplished by clicking the 'SearchEachFile' button after filling in the 'Files' field with the file names (or '@' followed by the name of a file that holds the list of file names). You can also specify a working directory for file lookup by filling in the 'Working Directory' field. When you LEFT-click SearchEachFile, a new viewer is created, and one-by-one the files will be loaded and searched until a match is found. The list of files will be updated when a match is found, so that when you are finished with one file you can LEFT-click SearchEachFile again to look for the next one. You can hit the 'Stop' button at the top of the EditTool to interrupt the search, but you should not try doing other operations while the search is in progress since it resets the selection each time it loads a file.

Occasionally you will want to apply certain operations to an entire set of files. You can do this by filling in the 'Operations' field and the 'Files' field, and then clicking 'DoForEachFile'. A new viewer will be created, and one-by-one the files will be loaded, selected, edited, and saved. No confirmation is required for the saves, so the entire process can go on without you. In fact, you should not try to do anything else while this is going on since the operations use the primary selection. The one exception is the 'Stop' button which you can hit to terminate the process.

For example, to replace the word "apple" with the word "banana" in a number of files in directory ///Users/YourName.pa/Play/, type "apple" for Target, "banana" for Replacement, "DoSubstitute" for Operations, a list of short file names for Files, and "///Users/YourName.pa/Play/" for Working Directory. Click the DoForEachFile button and find something else to do while Tioga makes the substitutions. Note: You probably want to select "Match Words Only", particularly if you are replacing "Scene" by "OBScene", so that no harm will be done if you process the same file twice.

*Operations via the Command Tool*

You can invoke a set of operations from a command tool as well as directly from the EditTool. When you run TiogaExecCommands, one of the commands it registers is DoTiogaOps which expects a command line containing operations in the same format as in the operations field. One possible application of this is to create a command file that

initializes various EditTool switches to your favorite settings. For example, you could use the following command to set up some of the search parameters: DoTiogaOps IgnoreCase MatchWords MatchPattern.

*Edit Commands*

This section of the EditTool contains buttons for all the basic edit commands. These are useful if you can't remember what keys to hit for an infrequent command or if you've set your user category to beginner or intermediate to filter out certain commands. Many of the buttons correspond to commands that have been documented above. However, a few of them are primarily of use in constructing edit macros and have not been mentioned before. For completeness, all the buttons will be given a brief description.

Modifying selections

CaretBefore – move caret to front of selection

CaretAfter – move caret to end of selection

CaretOnly – reduce selection to a caret

Grow – selection grows in hierarchy of point, character, word, node, branch

Document – select entire document

PendingDelete – make primary selection pending delete

NotPendingDelete – make primary selection not pending delete

MakeSecondary – make the primary selection become the secondary selection

MakePrimary – make the secondary selection become the primary selection

CancelSecondary – remove the secondary selection

CancelPrimary – remove the primary selection

Copy, Move, and Translate

ToPrimary – copy/move secondary to primary

ToSecondary – copy/move primary to secondary

Transpose – transpose the primary and the secondary

Moving the caret

GoToNextChar – make primary caret only and move one character toward end of document or one toward start if you RIGHT-click instead of LEFT-clicking

GoToNextWord – like GoToNextChar, but move toward end by 'words' or one toward start if you RIGHT-click instead of LEFT-clicking

GoToNextNode – move caret one node toward end or one toward start if you RIGHT-click instead of LEFT-clicking

Saving and Restoring the selection

SaveSel-A – save primary selection to restore later

RestoreSel-A – restore the selection previously saved by SaveSel-A

SaveSel-B – save primary selection to restore later

RestoreSel-B – restore the selection previously saved by SaveSel-B

Extend the selection to matching brackets

(...) – extend primary selection to matching parens
<...> – extend to matching angle brackets
{...} – extend to matching curly brackets
[...] – extend to matching square brackets
"..." – extend to matching double quotes
`...` – extend to matching single quotes
-...- – extend to matching dashes
▶...◀ – extend to matching placeholder brackets

Find placeholders

Next▶...◀ – move primary selection to next placeholder
Prev▶...◀ – move to previous placeholder

Delete and Paste

Delete – delete the primary selection
Paste – insert saved text at caret
SaveForPaste – save text for later pasting; overwrites text saved by Delete

Repeat and Undo

Repeat – repeat the last command sequence
Undo – undo the last command sequence

Deleting character/word next to caret

BackSpace – delete the character to the left of the caret
BackWord – delete the word to the left of the caret
DeleteNextChar – delete the character to the right of the caret
DeleteNextWord – delete the word to the right of the caret

Inserting matching brackets around the selection

Add( ) – insert parens around the selection
Add< > – insert angle brackets
Add{ } – insert curly brackets
Add[ ] – insert square brackets
Add" " – insert double quotes
Add' ' – insert single quotes
Add- - – insert dashes
Add▶ ◀ – insert placeholder brackets

Capitalization

AllCaps – make the selection upper case
AllLower – make the selection lower case
FirstCap – capitalize each word in the selection and make the other letters lower case
InitialCaps – capitalize the first word in the selection and make the other letters lower case

Special characters

MakeOctalCharacter – convert the 3 digits before the caret to the corresponding character

MakeControlCharacter – convert the character before the caret to a control character

UnMakeOctalCharacter – convert the character before the caret to 3 digit representing its character code

UnMakeControlCharacter – convert the control character before the caret to a normal character

Tree structure commands

Break – break node at caret

Join – join caret node to one before it

Nest – move selection deeper in tree

UnNest – move selection higher in tree

Miscellaneous commands

CommentNode – set Comment property of all selected nodes to TRUE

NotCommentNode – set Comment property of all selected nodes to FALSE

ExpandAbbreviation – expand the abbreviation to the left of the caret

MesaFormatting – add Mesa looks, formats, and style to selection

Command 0 1 2 3 4 5 6 7 8 9 – do saved command macro

*Writing Edit Macros*

A few examples should get you started writing your own edit macros. First, assume you find yourself doing a lot of edits of the form <stuff> becomes <pred>[<stuff>] for various values of <stuff> and <pred>. You might like a single command to insert the brackets and move the caret to the place where you will insert the <pred>. To construct such a command, first select a particular <stuff>, hit the Add[ ] button in the EditTool, the CaretBefore button, and the GoToNextChar button using the right mouse button. Then hit GetLast to fill the Operations field, enter '1' in the 'Command [0..9]' field, and hit SetCom to save the macro definition. Now you can select <stuff>, hit CTRL-1, and be ready to insert before the left bracket.

A macro to delete matching parentheses will show the use of the SaveSel and RestoreSel commands. Make a selection anywhere inside a pair of matching parens. Give the following sequence of commands: (...) to extend the selection, CaretAfter to position the caret at the right, SaveSel-A so we can restore the selection later, BackSpace to delete the right paren, RestoreSel-A to get the selection back, CaretBefore to move the caret to the front, and finally DeleteNextChar to delete the left paren. Now you can hit GetLast and SetCom to save this macro.

Other operations on the EditTool can also be programmed using edit macros. For example, assume you want a macro to substitute 'which' for 'that'. The following set of commands will load the target and replacement fields and do the substitute: RIGHT-click to select and clear the target field, type 'that' as the target text, RIGHT-click to select and clear the replacement field, type 'which' as the replacement text, and finally hit the Substitute button. This macro will only change the target and replacement fields. The

other parameters of the substitute are not changed and thus behave like 'free variables' of the macro. If you want to bind more of the choices, such as Match Case or Ignore Case, you simply include those commands as part of the macro. For example, you could select Ignore Case just before restoring the selection, and the macro would always set the ignore case flag when it was executed. Note that you must hit the 'Target' and 'Replacement' buttons to enter the target and replacement text. Directly selecting in those fields would not produce a correct macro because it would terminate the edit sequence and would also fail to identify which field was being filled in.

When you save an edit macro under a CTRL-number key, it only stays around for the rest of the session. If you want to save one for tomorrow, you can of course copy the operations to a file and redefine it another time. But if you really want to make a macro a permanent part of your user interface, you can add it to your 'TIP table' which describes the translation from keyboard and mouse actions into executable tokens such as those in edit macros. The details of how to do this are given in a later section.

## The EditHistory Tool

The EditHistory tool lets you undo edits in the same way that Undo does, but it lets you go back farther in history than just the most recent sequence. To create an EditHistory tool, type 'EditHistory' to the CommandTool. At the top of the tool are two fields and four commands. The bottom part of the tool is a text field to hold descriptions of previous edit events.

The system keeps a history of a certain number of the most recent edit events. You can find out how large this history is by the 'Get' command which will enter a number in the 'history size' field. The 'Set' command will change the history size to whatever value you've entered in the size field. The default size is 20; you can change this by making an entry in your User.Profile of the form 'EditHistory: <history-size>'.

The 'Show' command will display the events starting with the number in the 'since event number' field. If that field is empty it will show as many as are still remembered. The format of the entries is <event-number>, TAB, and then a list of operations. The 'Undo' command undoes the edits since the specified event number.

## Printing

The preferred route for printing is to create an Interpress master, and to send this to a printer; consult CedarChest documentation for details. Currently the command for doing this translation is called TiogaToInterpress, obtained via TiogaImager.df.

In the glorious future, Tioga could conceivably have an interactive typesetter that will let you make incremental revisions to a typeset document to adjust pagination, page layout, and other details before actually printing it.

## TIP Tables for Tioga

The acronym 'TIP' stands for 'terminal interface package'. TIP tables describe the translation from keyboard and mouse actions into executable tokens. The standard TIP table for Tioga is found in the file 'Tioga.TIP' and contains all the gory details about things such as multi-clicks of mouse buttons with various combinations of shift keys. It's unlikely that you want to try changing anything in Tioga.TIP, but you can consider adding commands to your own TIP table for things such as your favorite set of edit macros. Your TIP table can be 'layered' on top of the Tioga table so that the system will try to interpret actions according to your table before looking at the standard one. The User.Profile contains an entry named 'Tioga.TiogaTIP' in which you can enter the file name of your TIP table to be layered over the standard Tioga table. The entry should look like something like this

**Tioga.TiogaTIP**: MyOwn.tip Default

where 'MyOwn.tip' is the file name for your table. Notice that your table goes first in the list; definitions in tables occurring early in the list take precedence over those that appear later. The special name 'Default' refers to Tioga's default TIP table and will typically be the last entry in the list.

There are corresponding entries **Tioga.ReadonlyTiogaTIP** and **Tioga.TypescriptTIP** that apply to readonly documents and typescript documents, respectively.

The file 'MyOwn.tip' contains a sample table that you can edit to produce your own. It also contains documentation about the syntax of TIP tables and the macro package which is used with them.

Use the command ReadTiogaTipTables to reload a TIP table after you've changed it, or, if your user category is advanced, hit CTRL-! (really CTRL-SHIFT-1) to reload the Tioga profile information.

## Styles

A style is a collection of interpretations for looks and formats. The interpretations are represented as procedures written in a simple language that uses the JaM interpreter. The procedures set various formatting parameters such as font size and line spacing.

Note: If a document doesn't specify a style, the system will use a default. You can say what the default will be by adding user profile entries for DefaultStyle or ExtensionStyles (described below).

To determine the formatting parameters for a particular node, the system first gets the parameters for its parent and then executes the appropriate formatting procedure from the style. If the node doesn't have an explicit format, or if the format it has is not defined in the style, the system will execute the default formatting procedure instead. For root nodes, the default is the rule named 'root'; for other nodes, it is the rule named 'default'.

Formatting parameters for the text within a node are determined in a similar manner. The system first gets the parameters for the node and then executes the formatting procedures for the looks of the text. For look 'a', it executes the rule named 'look.a', for look 'b', the rule named 'look.b', etc.

### Properties related to styles

*Prefix and Postfix properties*

The values of these properties are command sequences that might be part of a formatting procedure. The Prefix commands are executed just before the standard formatting procedure for the node, and the Postfix commands are executed just after it. This makes it possible to modify the values of the formatting parameters that are the input and output of the format. You may want to use this to make local formatting changes, such as modifying tab stops for a particular table. However, don't abuse this facility. If you find you are making many local changes, you should probably modify the style instead.

*StyleDef property*

In typical use, a style lives in its own file and is referred to by the various documents that use it. However, in some cases you may have a style that is only used in one document, and you'd like to include it as part of the document to avoid the trouble of maintaining the style as a separate file. You can do this by using the StyleDef property. The value of this property is a style definition without any node structure or formatting looks. The style is automatically saved and loaded as part of the file and applies to the node and its children just as if you had set the node style in the usual manner with the EditTool.

[Note: If you use a StyleDef and it doesn't seem to do anything, do a Clear of the style name using the EditTool.]

## Style definitions

A style definition begins with the command 'BeginStyle' and ends with 'EndStyle'. Between these come any number of commands and definitions. The style is parsed and interpreted using JaM, so you will want to read the JaM documentation if you're going to spend much time writing styles.

*Attached styles*

You may want to define a new style that is only slightly different than an existing one. One approach would be to copy the existing style and edit it to make the changes. However, it may be preferable to track whatever modifications to the other style that may happen in the future. This can be done by 'attaching' the old style to the new one. For example, if your new style includes a command of the form

(Cedar) AttachStyle

then the new style will automatically include everything from the current version of Cedar style. Thus, if a format is not defined in the new style, it will be taken from Cedar style instead.

*ScreenRules, PrintRules, and StyleRules*

The basic form for a rule definition in a style is

(name) "comment string" { commands } StyleRule

In many cases it will be desirable for a rule definition to be different depending on

whether the output is for printing or for display. To simplify this, you may define a format both as a ScreenRule and as a PrintRule. The ScreenRule definition will be used for display, while the PrintRule will be used for printing. If there is only a StyleRule definition, it will be used for both printing and display.

*Formatting Parameters*

This section lists some of the formatting parameters. Those that are recognized by the style machinery but are not currently implemented are in ~~strikeout~~ ~~type~~. Typically, the name of the parameter is a command defined in JaM that pops an item from the stack and makes it the new parameter value. However if the item on the top of the stack is the word 'the', the commands push the current parameter value so procedures can read parameters as well as write them. For numeric parameters, the following mechanisms are provided to simplify making incremental changes to the current parameter value:

| | |
|---|---|
| <amount> bigger | adds the amount to the current parameter value |
| <amount> smaller | subtracts the amount from the parameter value |
| <amount> percent | multiplies the parameter value by the specified percentage, i.e.,<br>value ← (amount/100)*value |
| <amount> percent bigger | increases value by specified percentage |
| <amount> percent smaller | decreases value by specified percentage |

*Font Parameters*

There are more parameters here than can reasonably be referenced using the limited number of looks available in a given style. The Postfix property may be applied to individual characters to get fine-grain control of these parameters when you run out of looks, or for special cases. See above for instructions about how to do this.

| | |
|---|---|
| ~~alphabets~~ | one of *caps+lowercase*, *caps+smallcaps*, *lowercase*, or *caps* |
| fontPrefix | prefix of the full hierarchical font name, e.g. "xerox/pressfonts/" |
| family | the name of current font family, such as "TimesRoman" |
| size | value is the font size in points |
| face | one of regular, **bold**, *italic*, or ***bold + italic*** |

You can also add or remove italic or bold by means of the following commands:
+**bold face**, -bold face, +*italic face*, or -italic face

| | |
|---|---|
| textHue | hue of text, in range [0..1):<br>0=red, 1/6=yellow, 2/6=green, etc. |
| textSaturation | saturation of text: 0=no color, 1=full color |
| textBrightness | brightness of text: 0=black, 1=full brightness |
| textColor | takes 3 numbers: hue, saturation, brightness |
| underlining | one of *all*, *visible*, *letters+digits*, or *none* |

| | |
|---|---|
| underlineThickness | thickness of the underline. |
| underlineDescent | distance from baseline downwards to top of underline |
| underlineHue, underlineSaturation, underlineBrightness, underlineColor | parameters describing the color of the underline. |
| strikeout | one of *all*, *visible*, *letters+digits*, or *none* |
| strikeoutThickness | thickness of the ~~strikeout~~ line. |
| strikeoutAscent | distance from baseline to top of ~~strikeout~~ line. |
| strikeoutHue, strikeoutSaturation, strikeoutBrightness, strikeoutColor | parameters describing the color of the ~~strikeout~~ line. |
| backgroundAscent, backgroundDescent, backgroundHue, backgroundSaturation, backgroundBrightness, backgroundColor | parameters describing the size and color of the **text background**. |
| outlineBoxBearoff | amount to fatten text background horizontally, or, if no background box is specified, amount of extra white space to leave around the font box, both horizontally and vertically. |
| outlineBoxThickness | thickness of outline around text background. |
| outlineboxHue, outlineBoxSaturation, outlineBoxBrightness, outlineBoxColor | parameters describing the color of the **text outline box** |
| letterspacing | space to add between visible characters |
| hShift | distance to offset text, positive to the right |
| vShift | distance to raise text above baseline (can be negative) |

## *Fancy Tabs*

The style machinery actually allows for a fairly elaborate specification of fancy tabs, but only a tiny subset of this is actually used by the formatter. What is documented here is what actually works.

| | |
|---|---|
| clearTabStops | remove fancy tab stops |
| flushLeft tabStop | define a new tab stop. The only form of fancy tabs currently implemented is of this form. |

Example: to set up tab stops for old-fashioned assembly code, use something like

```
clearTabStops
10 sp flushLeft tabStop
16 sp flushLeft tabStop
35 sp flushLeft tabStop
72 sp flushLeft tabStop
```

In many cases, you will want to use an actual unit instead of "sp".

This lets you do silly things like

```
LOOP CR R1, R2   TEST THE LOOP COUNT 00100
     BH OUT      EXIT IF DONE        00200
     LA R1, 1(R1) NEXT BYTE          00300
OUT EQU *                            00400
```

*Graphics Parameters*

~~areaHue, areaSaturation, areaBrightness, areaColor, outlineHue, outlineSaturation, outlineBrightness, outlineColor~~

~~lineWeight~~

*Leading Parameters*

Leading parameters are stored as triples of <size, stretch, and shrink> which, following Knuth, we refer to as 'glue'. You can set the separate components individually, or you can push three values on the stack and use one of the leading glue commands to set them all at once.

There are three kinds of leading corresponding to the spaces between lines in a node, the space above a node, and the space below it. The actual space between two adjacent nodes is the maximum of the 'below' leading of the first node and the 'above' leading of the second.

| | |
|---|---|
| leading | distance between baselines within a node |
| leadingStretch | how much leading can increase |
| leadingShrink | how much leading can decrease |
| leadingGlue | size, stretch, and shrink for leading |
| topLeading | distance between baselines above a node |
| topLeadingStretch | how much top leading can increase |
| topLeadingShrink | how much top leading can decrease |
| topLeadingGlue | size, stretch, and shrink for top leading |
| bottomLeading | distance between baselines below a node |
| bottomLeadingStretch | how much bottom leading can increase |
| bottomLeadingShrink | how much bottom leading can decrease |
| bottomLeadingGlue | size, stretch, and shrink for bottom leading |

*Layout Parameters*

| | |
|---|---|
| lineFormatting | FlushLeft, FlushRight, Justified, Centered |
| lastLineFormatting | FlushLeft, FlushRight, Justified, Centered: applies to last line of node. |
| hyphenation | Takes a name (i.e., (English) or (No)) that tells what hyphenation routine to use (applies at node |

| | |
|---|---|
| | level). |
| hyphenCode | Takes a numeric character code specifying what to use for a hyphen (default 45 is the ASCII hyphen/minus). |
| maxHorizontalExpansion | Spaces in justified lines are never stretched by more than this amount, possibly resulting in a somewhat ragged right margin. Applies at the node level. |
| ~~minLineGap~~ | min distance between line top and previous bottom (can be negative) |
| leftIndent | all lines indent this much on left |
| rightIndent | all lines indent this much on right |
| firstIndent | first line indents this much more on left |
| firstIndentRight | first line this much shorter on right |
| restIndent | other lines indent this much more on left |
| ~~runaroundLeft~~ | depth of first indent on the left |
| ~~runaroundRight~~ | depth of first indent on the right |
| topIndent | top baseline at least this much down from top of viewer/page |
| bottomIndent | bottom baseline at least this up from bottom of page |
| lineLength | maximum width of lines of text |

*Page Layout Parameters*

| | |
|---|---|
| column | number of columns |
| pageWidth | width of the paper |
| pageLength | height of the paper |
| leftMargin | whitespace at left of the page |
| rightMargin | whitespace at right of the page |
| topMargin | whitespace at top of the page |
| bottomMargin | whitespace at bottom of the page |
| headerMargin | height of area below topMargin reserved for headers |
| footerMargin | height of area above bottomMargin reserved for footers |
| bindingMargin | extra margin on the bound edge of page |
| ~~pageBreakAfterFirstLinePenalty~~ | |
| ~~pageBreakAfterLastLinePenalty~~ | |

~~pageBreakBeforeFirstLinePenalty~~

~~pageBreakBeforeLastLinePenalty~~

~~pageBreakPenalty~~

*Dimensions*

| | |
|---|---|
| pt | point (72.27 pt/in) |
| bp | big point (72 bp/in) |
| pc | pica |
| in | inches |
| cm | centimeters |
| mm | millimeters |
| fil | 10↑4 points |
| fill | 10↑8 points |

*Miscellaneous*

| | |
|---|---|
| style | the name of the current style |
| isComment | pushes .true or .false according to comment property of node |
| isPrint | pushes .true if executing print rules, else .false |
| nestingLevel | pushes integer: 0 for root, 1 for top level, etc. |

## Tioga Commands

### ReadTiogaTipTables

Causes Tioga to read its TIP tables again. If your user category is advanced, you can also invoke this operation by selecting in any Tioga document and hitting CTRL-!.

### WritePlain

Make Tioga files unformatted by eliminating everything except the plain text. Inserts leading tabs before each node according to its nesting in the tree and terminates each node with a carriage return.

### WriteMesaPlain

Same as WritePlain, except inserts double dashes at start of comment nodes.

### ReadIndent

Build Tioga files with one node per line of source with indenting based on white space

at the start of lines.

**TiogaMesa**

Convert Mesa files to Tioga format by combining a ReadIndent with a CTRL-M over the entire file.

**DoTiogaOps**

Expects a command line containing operations in the same format as in the EditTool operations field. Among other things, you can use this to initialize various EditTool choices such as IgnoreCase or MatchWords.

**AnnotateProperties**

Creates a system button, TiogaProps, to annotate Tioga documents with comments describing the node properties and the document nesting structure. LEFT-click TiogaProps with the selection in a document and the annotations are added; RIGHT-click TiogaProps and the annotations are removed from the selected document.

When a list of file names is given as command arguments, AnnotateProperties writes new versions of the files with the annotations in place.

**PruneAnnotations**

PruneAnnotations takes a list of files and writes new versions of the files with the annotations removed.

## Tioga User Profile Entries

**Default keep**

This determines the default number of versions to keep for files created by Tioga.

Tioga.defaultKeep: 2

**Default file extensions**

This determines what extensions Tioga should look for in opening files. The entry is of the form

Tioga.SourceFileExtensions: mesa tioga df cm config style

You may also have an entry for implementation extensions to be used with the 'load impl' commands.

Tioga.ImplFileExtensions: cedar mesa

**Default Styles Determined by File Extensions**

This entry specifies the default style to be used with documents that do not explicitly

name a style. The style is determined by the extension in the file name. The entry is of the form

Tioga.ExtensionStyles: <extension1> <stylename1> <extension2> <stylename2> ...

To specify a default style for files with no extension in their name, use the fake extension name 'null' in this list.

**Default Style**

This entry specifies the default style to be used with documents that do not explicitly name a style and do not have an extension given in the ExtensionStyles list. The entry is of the form

Tioga.DefaultStyle: Cedar

**Default submenus**

This entry specifies which menus, if any, should automatically be displayed when you create a new Tioga viewer by clicking one of the buttons in the upper right corner or by using the Open or New commands. The entry is of the form

Tioga.DefaultTiogaMenus: places levels

or

Tioga.DefaultTiogaMenus: none

You may delete or reorder the menu names to suit your tastes.

**Open First Level Only**

If set to true, documents will be opened with only their first level showing. Default is false.

Tioga.OpenFirstLevelOnly: TRUE

**Style Search Rules**

This controls how Tioga uses the short style names to find files containing the style and abbreviation definitions. When a style is first requested, Tioga looks in these directories in the specified order for the first matching name that has the ".style" extension, and uses that. The same thing happens with abbreviations, except the ".abbreviations" extension is used instead. Tioga will notice when this name-to-file binding changes, and automatically reload the appropriate internal tables.

Tioga.StyleSearchRules: []<>7.0>Commands> []<>7.0>System>

**Scroll bottom offset**

When you are typing and the caret goes to a new line just off the bottom of the viewer, Tioga will automatically scroll the viewer up a little to make the caret visible again. This parameter controls how far up to scroll: a big number causes larger but less frequent glitches.

Tioga.ScrollBottomOffset: 3

### Scroll top offset

When you do a Find command you may want to see a few lines in front of the match to give you more context. This parameter tells Tioga how many extra lines to want in such situations. The entry is of the form

Tioga.ScrollTopOffset: 1

### Selection Caret

The default behavior for Tioga is to place the caret at the end nearer the cursor when the selection is made. Some people have requested to have the caret always placed at one end or the other, hence this profile entry. The choices are before, after, and balance. The entry is of the form

Tioga.SelectionCaret: before

### Selection Displacement

This lets you specify a vertical displacement for making selections. Tioga behaves as if you had pointed this number of points higher up the screen so that you can point at things from slightly below them. The entry is of the form

Tioga.YSelectFudge: 2

### TIP Table

This entry specifies the TIP tables to use with Tioga documents. The entry is of the form

Tioga.TiogaTIP: MyOwnTip.tip Default

See the section on TIP tables for more information.

### Unsaved Documents Cache Size

This controls the number of unsaved documents the system will remember. The entry is of the form

Tioga.UnsavedDocumentsCacheSize: 4

### Show Unsaved Documents List

If this is true, a viewer will be created holding an up-to-date list of the unsaved documents that can still be reloaded. The entry is of the form

Tioga.ShowUnsavedDocumentsList: TRUE

### User category

As described above, this entry lets you control your user category. The alternatives are Beginner, Intermediate, and Advanced. The entry is of the form

Tioga.UserCategory: Intermediate

**Version Map lookup**

If this is true, Get, GetImpl, Open, and LF will look for the requested file in the Cedar version map if their earlier attempts to open a file fail. The entry is of the form

Tioga.TryVersionMap: FALSE

## Command Summary

This is a short summary of the Tioga commands available using the keyboard and mouse. Other commands are available through the EditTool, the EditHistory Tool, and various command operations.

The extra keys on the keyboard are used for common editing actions. Recall that the bottom right blank key can be used as another CTRL key, and you can use either CTRL key and either SHIFT key interchangeably.

The word CLICK refers to the action of rapidly pressing, releasing, and repressing a key.

| | |
|---|---|
| ESC | Repeat last action |
| SHIFT-ESC | Undo last action |
| ESC-select | Automatic repeat of last action when finish selection |
| DEL | Delete |
| LF | Load file in 'No Name' viewer |
| CTRL-LF | Load Impl file in 'No Name' viewer |
| BS | Backspace character |
| SHIFT-BS | Delete next character |
| CTRL-BS | Backspace word |
| CTRL-SHIFT-BS | Delete next word |
| NEXT | Find next placeholder (middle blank key) |
| SHIFT-NEXT | Find previous placeholder |
| RETURN | Insert carriage return with leading spaces copied from previous line |
| SHIFT-RETURN | Insert carriage return |
| CTRL-RETURN | Break node |

Mouse button clicks are used for making selections. Selection granularities are point, character, word, node, branch, document.

| CLICKS | LEFT | MIDDLE | RIGHT |
|---|---|---|---|
| SINGLE | Select letter | Select word | Extend selection at current granularity |
| DOUBLE | Select node | Select branch | Reduce selection granularity & extend |
| TRIPLE | | Increase selection granularity & extend | |

Selections are used for delete, copy, and move.

| | |
|---|---|
| CTRL-select | Delete when finish selection |
| SHIFT-select | Copy to primary |
| CTRL-SHIFT-select | Move to primary |

LOOK commands are used for editing looks. (The LOOK shift is the top blank key.)

| | |
|---|---|
| LOOK-char | Add look to selection |
| LOOK-SHIFT-char | Remove from selection |
| LOOK-space | Remove all from selection |

Except for CTRL-A, CTRL-H, and CTRL-W, the following commands are not enabled for beginning users. We provide both CTRL-A and CTRL-H as a convenience to users with strong habits from previous systems. A '*' indicates a command enabled for advanced users only.

| | |
|---|---|
| CTRL-A | Backspace character |

| | |
|---|---|
| CTRL-SHIFT-A | Delete next character |
| CTRL-B | Insert matching placeholder brackets |
| CTRL-C | Lower case |
| CTRL-SHIFT-C | Upper case |
| CTRL-CLICK-C | First cap |
| CTRL-SHIFT-CLICK-C | Initial caps |
| CTRL-D | Select document |
| CTRL-E | Expand abbreviation |
| CTRL-F-select | Copy format to primary * |
| CTRL-H | Backspace character |
| CTRL-SHIFT-H | Delete next character |
| CTRL-I | Indent (does Break and Nest) * |
| CTRL-SHIFT-I | Unindent (does Break and UnNest) * |
| CTRL-J | Join nodes * |
| CTRL-K | Make control character |
| CTRL-SHIFT-K | Unmake control character |
| CTRL-M | Automatic MESA formatting |
| CTRL-N | Nest * |
| CTRL-SHIFT-N | UnNest * |
| CTRL-O | Make octal character |
| CTRL-SHIFT-O | Unmake octal character |
| CTRL-P | Paste |
| CTRL-Q-select | Copy looks to primary |
| CTRL-S-select | Copy primary to selected destination |
| CTRL-T | Time |
| CTRL-V | Select visible (expand to selection to blanks) |
| CTRL-W | Backspace word |
| CTRL-SHIFT-W | Delete next word |
| CTRL-X-select | Select for transpose with primary |
| CTRL-Z-select | Move primary to selected destination |
| CTRL-] | Select matching [..]'s (same for }, ), and >) |
| CTRL-[ | Add matching [..]'s (same for ", `, -, {, (, and <) |
| CTRL-! | Have Tioga read its TIP tables again * |
| CTRL-← | Set format to word before caret * |
| CTRL-SHIFT-← | Insert format name * |
| CTRL-\ | Set comment property TRUE * |
| CTRL-SHIFT-\ | Set comment property FALSE * |