TEDIT RELELEASE NOTES

By Ron Kaplan

This document was last edited in March 2025.

March 2025

Minor adjustments

Tedit now passes the `HOSTSTREAM` as an additional argument to an image object `IMAGEBOXFN`.

`TEDIT.FORMAT.HARDCOPY` now has an additional `QUIET` argument that suppresses the printing in the promptwindow of the number of pages and the destination file. The caller can update the window before or after the hardcopy.

The caller of `TEDIT.MAP.OBJECTS` can now specify in the `COLLECT?` argument the values that are to be collected. The possible values for `COLLECT?` are

`OBJECT`: The image objects are collected.
`CH#`: The character positions of the objects are collected.
`VALUE`: The values returned by applying function FN to each object are collected.
`FIRST`: The return is the list (ch# object value) for the first object that satisfies the predicate FN. Essentially a test to see if there are any such objects.
Any other non-NIL value: The (ch# object value) for the objects are collected, as before.

`TEDIT.SELPROP` now makes the `SET` property accessible, and values for the `SHADE` and `SHADEHEIGHT` properties can be changed.

Region prompt at window opening

Previously, when a region or window was not provided in the call to TEDIT, the user would be asked to sweep out a screen region before Tedit examined any of the properties of the intended target of the edit. No guidance was provided to the user that distinguished an editing session on, say, an empty document from a session on a pre-existing formatted document, or a Lisp source file. The order of operations has been changed so that an existing document is now opened before the window is constructed, and information derived from the document is used to prompt for an appropriate initial region size.

In the case of a formatted document, Tedit offers a ghost region whose width is is determined from the document's margins and whose height is enough to show the first 20 lines of the file. For a Lisp source file or individual function (e.g. from Gitfns comparisons or Filebrowser `SEE` or `EDIT` commands), the width is heuristically set to allow a sufficient number of characters in the default font to minimize the number of unintended line-wraps. The initial region for a `TDRIBBLE` stream is the width and height of the Exec window.

In each of these cases, the user is prompted only to locate an anchor point for a ghost region of the precomputed fixed size, no sweeping is requested (or allowed). (The downsize of open-first/windowsecond is that if something goes wrong in the opening, the error shows up in the System promptwindow, not the promptwindow of an already open but empty window.)

New architecture for edit-character bindings

This release defines and implements a new architecture for separating the definition and implementation of editing actions from the specifications that bind particular characters to those actions. The possible editing actions are defined in the list `TEDIT.CHARACTIONS`.

`TEDIT.CHARACTIONS` [Variable]

Each element of `TEDIT.CHARACTIONS` is a pair of the form

```
        (ACTIONNAME IMPLEMENTATION)
```

where `ACTIONNAME` is an atom that identifies a particular action, for example `CHARDELETE`, `FIND.FORWARD`, `BOLD.ON`. The implementation for each action is either an atom or a list. An atom is the format that previously existed and is provided for backward compatibility. It is interpreted as a function of three arguments, the text stream `TSTREAM`, its `TEXTOBJ`, and its current selection `SEL`. A list is interpreted as a form to be evaluated in a context in which the variables `TSTREAM`, `TEXTOBJ`, `SEL`, and also a typed-in character code `CHARCODE` are bound as `SPECVARS`. This format allows more flexibility in the arguments that can be provided to the implementation and allows for the same implementation function to be parameterized with less clutter for different actions.

Here are some sample entries on `TEDIT.CHARACTIONS`:

```
        (SUBSTITUTE \TEDIT.CHAR.SUBSTITUTE)
        (NEST (\TEDIT.CHAR.NEST TSTREAM NIL))
        (UNNEST (\TEDIT.CHAR.NEST TSTREAM T))
```

The `SUBSTITUTE` action is implemented by applying the function `\TEDIT.CHAR.SUBSTITUTE` to the current selection of the text stream. The `NEST` and `UNNEST` actions (for indenting/unindenting both left and right margins) are implemented by calls to `\TEDIT.CHAR.NEST` that differ in the flag that indicates whether margins are to be brought in or pushed out.

The entries on `TEDIT.CHARACTIONS` are "built-in" in the sense that they are typically implemented through internal, private interfaces. But whether or not those actions are invoked by one or more keystrokes depends on whether they are bound to particular character codes. The binding of characters to actions is accomplished by installing a list of character bindings. This is intended as amore public interface, depending on personal preferences. But the default set of character bindings is provided by the variable `TEDIT.CHARBINDINGS`.

`TEDIT.CHARBINDINGS` [Variable]

Each character binding is a list of the form

```
        (ACTIONNAME CHAR1 CHAR2 ...)
```

where `ACTIONNAME` is one of the actions defined in `TEDIT.CHARACTIONS` and each `CHAR`$_i$ is typically the name of a character (e.g. `Meta,S`) but possibly a raw character code. Among the entries currently included in `TEDIT.CHARBINDINGS` are:

```
        (SUBSTITUTE "Meta,s" "Meta,S")
        (FIND.BACKWARD "Meta,F")
        (FIND.FORWARD.AGAIN "Meta,g")
        (CHARDELETE.FORWARD "^W" "^U")
        (NEST "Meta,[")
```

Installing a character binding list has the effect of binding each of the characters to the implementation of the associated action.

A set of character bindings is installed into the general `TEDIT.READTABLE` or the readtable of a particular stream by the function `TEDIT.INSTALL.CHARBINDINGS`:

`(TEDIT.INSTALL.CHARBINDINGS CHARBINDINGS RDTBL [CHARACTIONS])` [Function]

`CHARBINDINGS` is a list of character bindings, defaulting to the value of `TEDIT.CHARBINDINGS`. `RDTBL` can be a read table or a text stream whose read table will be affected. `RDTBL` defaults to the global `TEDIT.READTABLE`. (The optional `CHARACTIONS` defaults to `TEDIT.CHARACTIONS`; it is provided as an argument so that sophisticated users can introduce their own special actions.)

The expression `(TEDIT.INSTALL.CHARBINDINGS)` is evaluated when Tedit is loaded. It installs the actions specified in `TEDIT.CHARBINDINGS` in the global `TEDIT.READTABLE`.

`(TEDIT.CLEAR.BINDINGS BINDINGS RDTBL)` [Function]

Removes the bindings in `BINDINGS` from `RDTBL`, removing all bindings if `BINDINGS` is `NIL`.

`(TEDIT.GET.ACTION CHARCODE BINDINGS)` [Function]

Returns the name of the action, if any, that `BINDINGS` associates with `CHARCODE` (a code or the name of a character name). `BINDINGS` can be a binding list or a read table.

`(TEDIT.GET.BINDING ACTION BINDINGS)` [Function]

Returns the character codes (not names) that are bound to `ACTION` in `BINDINGS` (a list or read table). The value is a character code if there is only one, otherwise a list of codes.

Note that the functions `TEDIT.SETFUNCTION` and `TEDIT.GETFUNCTION` exist as before. They operate directly on the implementation of a given code, not on the action whose name serves as an intermediary.

The new architecture replaces the capabilities provided by the earlier `TEDITKEY`, `TKDORADO`, and `EDITKEYS` lispusers packages, and it obsoletes those files.

The character bindings of `TKDORADO` are defined in the binding list `TEDIT.DORADO.CHARBINDINGS`; essentially these assign a set of meta-control characters to various formatting actions.

The sensible `TEDITKEY` operations are included in `TEDIT.CHARACTIONS` and `TEDIT.CHARBINDINGS` (mostly) makes the same character assignments.

`EDITKEYS`-like edit buttons are constructed by the function `TEDIT.BUTTONS.BUILD`:

`(TEDIT.BUTTONS.BUILD BUTTONSPEC TITLE NROWS  CHARBINDINGS)` [Function]

`where BUTTONSPEC` is a list of elements of the form

```
    (BUTTONLABEL ACTION1 [ACTION2])
```

The image of a key labeled with `BUTTONLABEL` is constructed for each element. If a button is selected and a Tedit window has the input focus, a character bound to `ACTION1` is inserted into the system buffer if the shift key is not down, otherwise a character bound to `ACTION2` is inserted.

`BUTTONSPEC` defaults to the list `TEDIT.BUTTONS.SPEC`, `TITLE` defaults to "Tedit Buttons", `NROWS` is 1, and `CHARBINDINGS` is `TEDIT.CHARBINDINGS`.

```
TEDIT.BUTTONS.SPEC                                          [Variable]
```

`Here are some of the entries on TEDIT.BUTTONS.SPEC:`

```
    (Bold BOLD.ON BOLD.OFF)
    (Italic ITALIC.ON ITALIC.OFF)
    (Case  UCASE LCASE)
    ((Strike- out) STRIKEOUT.ON STRIKEOUT.OFF)
```

A new item Buttons appears in the Tedit default menu. `(TEDIT.BUTTONS.BUILD)` is evaluated when that item is selected.

Tedit Fifth Round: November 2024

Bug fixes

Performance improvements in line updating when typing and especially printing (the important slow case remains: printing at the end of the stream, e.g. for transcript/dribble applications)

Fixed glitches in transitions between fat XCCS characters on either side of image objects.

Fixed copying (shift-select, COPYINSERT) from Tedit to non-Tedit TTY windows, and from non-Tedit sources (Clipboard, Sedit) to Tedit.

And many other issues...

Overhauls

- More work on eliminating X-pointer cycle between the TEXTSTREAM and the TEXTOBJ (STREAMHINT field). Some paths still remain (there is a define flaw in that the stream returned to the user is also used for internall for search and display).
- Scrolling/display
  - Fewer line-update subroutines, for maintainability
  - Improved caret placement with keyboard input
  - Improved line and caret placement at bottom of pane and end of document, with or without EOL's
  - Scrolling when window's left side is off-screen now updates correctly
  - Fixed glitches in reshaping
- Selection
  - Removed (x)pointer cycle through selection datatype. TEDIT.GETSEL returns a copy that includes a pointer to the text stream, not the TEXTOBJ. (part of deprecating TEXTOBJ as a public data structure).
- Tedit menus
  - Rework of button primitives, simplified and more public interface (see below).
- HELP and SHOULDNT replaced by ERROR, unless \TEDIT.THELPFLG.

The number of state-variables in a Tedit process has been reduced, a step towards eventually having all Tedit's run in a single process.

New functionality

Meta keys:

D/| bring up DINFO man page, like Sedit, for a selected word (for Lisp source files or functions, a "word" is an atom according to the file's reader environment)

O|o apply ED to current word (like open in Sedit)

F finds backwards (vs f = finds forward) (wildcard #/* matching is improved)

g finds again forward
G finds again backward

<|, move caret back one character, without deleting. (Delete key deletes backwards.)

>|. move caret forward one character, without deleting. (Del key deletes forward.)

R|r re-applies the last history operation at a new location (e.g. for multiple inserts in different places).

U undoes the last undone history action (meta-u undoes the last event, as before). (History now maintains multiple events, not just the last one).

The DEL key deletes 1 character forward.

Tedit properties (at open or via TEXTPROP):

READONLY property can have value QUIET instead of just T: Still suppresses attempted changes but doesn't put up a warning message.

APPEND added to allow updates only at the end of the stream, also with T/QUIET options.

DON'TUPDATE value T suppresses updating of the display while other changes are being made, NIL restores the updating.

NOTSPLITTABLE value T suppresses the split-region of the Tedit window so that the window cannot be split (or unsplit). NIL restores splitting possibilities.

HISTORY value OFF erases the current history at the next document-changing event and suppresses further recording, so that document changes cannot be undone. ON restores history recording for future undoing. CLEAR discards current history and begins a new epoch of events. HISTORY is turned OFF by default when READONLY the APPEND properties are applied.

FILENAME returns the filename associated with the document. Cannot be set.

PAGEFORMAT returns or sets (by calling TEDIT.PAGEFORMAT) the page format for the document (cf TEDIT.SINGLE.PAGEFORMAT, TEDIT.COMPOUND.PAGEFORMAT).

DIRTY marks the document as dirty (T) or clean (NIL), changes titlebar *.

LENGTH (fetch only) returns the current document length in characters.

PARABREAKCHARS is a list of characters that cause paragraph breaks, usually (EOL CR LF). Setting to NIL means that no paragraph breaks will be inserted (and if there are no other character-looks changes, all characters will go into one extended piece). NIL is the best (and default) setting for Lisp source files, since there is no meaningful difference between paragraphs and lines.

The value of the FONT property can be a fontclass atom (e.g. DEFAULTFONT), no need to wrap it in FONTCREATE.

(TEDIT.SELPROP SEL PROP NEWVALUE) gets and sets the properties of a selection as returned by TEDIT.GETSEL, in order to hide Tedit internals. The following properties are valid:

CH# CHLIM LENGTH (=DCH) POINT KIND POINTCH# SELOBJ TEXTSTREAM SHADE SHADEHEIGHT

The selection returned by TEDIT.GETSEL is a copy of the document's current selection, with a pointer to the textstream as its TEXTSTREAM propery. The modified selection can be passed back in through public selection-taking functions (TEDIT.INSERT, TEDIT.SETSEL...).

String-like access to Text streams:

(TEDIT.NCHARS TSTREAM) [Function]
Returns the number of characters in the stream

(TEDIT.NTHCHARCODE TSTREAM N) [Function
Returns the Nth character-code of TSTREAM, NIL if N is out of bounds. Negative N counts from the end. If the Nth "character" is an image object, that object is returned. TSTREAM may be a selection (from TEDIT.GETSEL), in which case N picks from the characters in that selection.

(TEDIT.NTHCHAR TSTREAM N) [Function
The Nth character code of TSTREAM is returned as the corresponding character.

(TEDIT.RPLCHARCODE TSTREAM N NEWCHARCODE NEWCHARLOOKS DONTDISPLAY) [Function]
Replaces the character at position N with NEWCHARCODE (which may be an image object). NEWCHARLOOKS can specify the looks of the new character if they are to be different from the looks of the character being replaced. Does not extend the stream: N must be between 1 and the length of the stream. (BOUT extends).

(TEDIT.CONCAT TSTREAMS SEPARATOR) [Function]
Returns a textstream containing the concatenation of the ordered list of TSTREAMS, separated by SEPARATOR if provided. SEPARATOR can be a character name (e.g. FORM), a character code, or a string. The specification for page numbering is taken from the first TSTREAM and carries through.

(TEVAL FORM WINDOW TITLE DONTDEFER) [Macro]
Evaluates FORM with the TTY stream set to a text stream in WINDOW with title TITLE. History events are not recorded, and the stream is initialized as APPEND QUIET. Screen updating is suppressed until the evaluation of FORM is complete, unless DONTDEFER. At that point the stream is changed to READONLY QUIET. Example: (TGREP --) is implemented as (TEVAL (GREP --). (Note that TTY keyboard interaction won't be visible unless DONTDEFER; TDRIBBLE may be more appropriate). The PARABREAKCHARS property is set to NIL.

(TEDIT.DEFER.UPDATES TSTREAM AFTERPROPS) [Function]
Suppresses display updating of TSTREAM until the exit of an enclosing RESETLST. At that point AFTERPROPS are applied to the stream. This is a simple way of avoiding the

perceptible slowness of incremental screen updates when a program is filing up a textstream.

(TDRIBBLE) [Function]
Dribbles to a text stream that opens in a scrollable/searchable window when dribbling is ended by a call to DRIBBLE. The stream is APPEND QUIET with PARABREAKCHARS NIL.

(TEXTSTREAM TSTREAM NOERROR) [Function]
If the new argument NOERROR is T, returns NIL if TSTREAM cannot be coerced to a text stream, otherwise causes an error as before.

Changed behavior

Textstream BOUT now behaves like the generic BOUT (and character printing) to other output streams: appends at the end, replaces in the middle.

BOUT creates a history event by default, so it is undoable and the history keeps in step with the state of the document. A sequence of BOUT's appended to the end of the document will be collected into a single event and undone all together. BOUTs that replace characters in the earlier in the document will be recorded as separate events. History will be suppressed if the text property HISTORY is set to OFF.

[Soon: History is not cleared after a Put is performed, so that it is possible to back out of edits made before a checkpoint file has been written.]

Line leading has been moved from the line descent to the line ascent. The effect is that highlighting stays close to the bottom of a selection's line rather than close to the top of the line below. The difference is visually apparent if the leading is large, for example, double spaced.

The TO and FROM selection-arguments of TEDIT.MOVE and TEDIT.COPY can be specified as text streams as well as selections. The source and target are then taken as the current selections of those streams. This may make it unnecessary in some cases to use TEDIT.GETSEL to extract a free-floating selection, which should be discouraged.

(More) public support for Tedit menus

There is a new, cleaner, more maintainable, and more public platform for creating and interpreting Tedit menus (like Character Looks, Paragraph Looks, etc.). These functions are in the new file TEDIT-BUTTONS and the code in TEDIT-MENU shows how they are used. Generally the function in TEDIT-BUTTONS have names prefixed with MB (for Menu Button).

(MB.ADD MENUDESC MENUTSTREAM WHERE) [Function]
MENUDESC is a list of descriptors of menu items to be inserted as a sequence in the pre-existing textstream MENUTSTREAM. The first menu-item will be inserted after WHERE if provided, otherwise at the end of the stream.

A menu-item descriptor can be

A string, presumably just a rubric whose characters will be inserted. The atoms EOL and TAB stand for the strings containing those singleton characters. The string's characters will be inserted in the stream's current font. The characters will be protected: not selectable.

A positive integer: that many protected spaces will be inserted.

A list of the form (TEXT (STRING string) (FONT fontspec)). This is treated like a string except that the characters are in fontspec, a font (e.g. (HELVETICA 10 BOLD)), a font descriptor, or a font class.

A list of the form (Buttontype . buttonproperties) that specfies a button image object of the indicated type with the specified properties.

Among the properties are at least one of IDENTIFIER and LABEL. The function MB.GET locates the button by its litatom IDENTIFIER, LABEL is the string that is displayed for the button (except that FIELD buttons have PRELABEL and POSTLABEL properties). If only LABEL is provided, IDENTIFIER is assumed to be (U-CASE LABEL). If only IDENTIFIER is provided, it is also taken to be the label.

Most buttons also have:

STATE The result of clicking on the button.

INITSTATE The initial value of STATE

FONT The font for the button's LABEL display, if not (HELVETICA 8 BOLD).

SELECTFN A function of the object, the menu's window, selection, and the textstream. This is executed in the window's coordinate system, if the button is selected (buttons come up while mouse is in the button).

STATECHANGEFN A function called to take special action whenever the STATE changes. Arguments: button-object, new state, menu-stream.

SETSTATEFN A function called to install a new state. Arguments: the piece containing the button, a new value, and menu-stream. The piece permits searching for the state of other buttons before or after (e.g. The Other font toggle looking for the field containing the new font-family or the font-family object where the new family will be installed. The given value may be something that should be coerced to ON or OFF (e.g. Bold is turned ON if the font of a document's current selection is bold).

GETSTATEFN A function that may map the state as determined by the button type into some other interpretable value or action.

IGNORE T if a button does not represent an independent characteristic of the character, paragraph, or page looks. For example, TABTYPE indicates how to interpret tabs as they are specified in the tab-stop bar of the paragraph menu, but it is not an independent property of the paragraph looks.

The button types and their further relevant properties are:

ACTION [Button type]
Executes the SELECTFN when selected. The APPLY, SHOW, and NEUTRAL buttons are implemented as ACTION buttons with appropriate SELECTFNs. The button is highlighted (inverted) while the mouse is tracking, then unhighlighted when the mouse buttons come up and the function is executed.

TOGGLE [Button type]
Toggles the STATE between ON and OFF. The button is displayed in inverse video when the button is ON. INITSTATE defaults to OFF.

3STATE [Button type]
Like TOGGLE, but switches between ON (label is inverted), OFF (label has a strike-out line), and NEUTRAL (label is not inverted).

NWAY [Button type]
Defines a collection of mutually exclusive toggles. When one internal button is turned on, the current ON button is turned OFF. The button's STATE is the identifier of the current ON button, if any.

FIELD [Button type]

Defines a field between { and } for the user to fill in. A FIELD button has additional properties:

PRELABEL A string to appear, protected, before the field-opening {

POSTLABEL A string to appear after the field-closing }.

LABELFONT Font for the pre and post labels.

FIELDTYPE One of ATOM STRING NUMBER POSITIVENUMBER CARDINAL IMAGEOBJ.

FIELDFONT Font for the field characters.

If a field is empty, its value is determined by the EMPTYVALUE property. If that property is not specified, NIL is returned for empty ATOM fields, the empty string "" is returned for empty STRING fields. and the atom **EMPTY** is returned for other field types.

When the mouse is clicked in either the PRELABEL or POSTLABEL of a field button, the cursor moves and flashes inside the following/preceding bracket, for ordinary Tedit input.

(MB.DELETE IDENTIFIERS MENUSTREAM) [Function]

Deletes the buttons for all IDENTIFIERS in MENUSTREAM.

(MB.GET IDENTIFIERS MENUSTREAM RETURNS BEFORE START) [Function]

Returns information about all the IDENTIFIERS in MENUSTREAM, searching before START if BEFORE, otherwise after. START is a menu selection, a menu piece, a menu character number, or NIL (= end of the document if BEFORE, otherwise its beginning).

If IDENTIFIERS is an atom, the value is information about just that button. Otherwise it is a plist keyed by the identifiers in IDENTIFIERS.

RETURNS specifies what kind of information is returned. It can be any of OBJECT STATE STARTCHNO ENDCHNO STARTPC ENDPC, or a list of those specifiers. If a list, the value will contain for each identifer a list of corresponding items. RETURNS defaults to OBJECT, and ALL is a shorthand for (STATE OBJECT STARTPC ENDPC STARTCHNO ENDCHNO).

The value for any of the identifiers will be NIL if such a button cannot be found.

Debugging tools: internal/TEDIT-DEBUG

This file contains an assortment of tools for visualizing different Tedit data structures and testing different capabilities. It is not yet documented and new functions get added as the work proceeds and old ones are revised. The tools are available for poking around after LOAD(TEDIT-DEBUG.LCOM), but then executing (TEDIT-DEBUG) will set up the environment for more serious debugging.

To give a feeling for what is available, the visualization functions typically begin with the letter S (for "show"), followed by a short/cryptic (because they are typed often) indication of what is to be shown. Thus (SL) shows the lines in the currently focused Tedit window or stream, SP shows its pieces, SLF displays a parse of the looks-portion of a formatted file, etc.

If there is only one Tedit on the screen, that becomes the focus. Otherwise, the focus defaults to the Tedit that the mouse is hovering over, and the focus stays that way until one of these functions is executed with the mouse over a different Tedit.

The output goes to the exec window by default, but for some functions the output can be directed to a separate Tedit stream. Thus (SP NIL T) will show all the pieces of the current stream in a new Tedit window that is scrollable, searchable, and copy-selectable.

Functions beginning with I (for "inspect") will bring up the inspector on various datastructures.