TextEdit Reference

(Not Recommended)

Carbon > Text & Fonts

2006-07-13

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

TextEdit Reference (Not Recommended)

Framework: Carbon/Carbon.h

Declared in TSMTE.h
TextEdit.h

Important: The information in this document is obsolete and should not be used for new development.

Overview

TextEdit was originally designed to handle editable text items in dialog boxes and other parts of the Mac OS system software. Although TextEdit was enhanced to provide more text-handling support, especially in its handling of multi-script text, it retained some of its original limitations. TextEdit was never intended to manipulate lengthy documents or text requiring more than rudimentary formatting.

TextEdit has been deprecated for deployment targets Mac OS X version 10.4 and later. The replacement API is Multilingual Text Engine (MLTE). MLTE offers additional features such as Unicode text editing, document-wide tabs, full justification of text, support for more than 32 KB of text, built-in scroll bar handling, built-in printing support, support for inline input, support for the advanced font features of Apple Type Services for Unicode Imaging (ATSUI), and support for multiple levels of undo.

You should use MLTE to replace TextEdit functions in your existing applications. With MLTE, you can significantly reduce the number of lines in your code because MLTE handles most of the low-level tasks you had to code in the past. MLTE provides a quick and easy solution for static display of Unicode text and for creating Unicode-compliant text-editing fields within an application. For more information, see *Handling Unicode Text Editing With MLTE*.

Functions by Task

Activating and Deactivating an Edit Structure

`TEActivate` (page 71) Deprecated in Mac OS X v10.4

Activates the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEDeactivate` (page 77) Deprecated in Mac OS X v10.4

Deactivates the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Using Additional TextEdit Features

`TEFeatureFlag` (page 79) Deprecated in Mac OS X v10.4

Turns a specified feature on or off or returns the current status of that feature. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Checking, Setting, and Replacing Styles

`TEContinuousStyle` (page 74) Deprecated in Mac OS X v10.4

Determines whether a given character attribute is continuous over the current selection range. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetStyle` (page 85) Deprecated in Mac OS X v10.4

Gets character attributes for the specified text. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetStyleScrapHandle` (page 86) Deprecated in Mac OS X v10.4

Creates a style scrap structure, copies the character attributes associated with the current selection range into it, and returns a handle to it. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TENumStyles` (page 90) Deprecated in Mac OS X v10.4

Returns the number of character attribute changes contained in the specified range, counting one for the start of the range. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEReplaceStyle` (page 92) Deprecated in Mac OS X v10.4

Replaces any character attributes in the current selection range that match the specified existing character attributes with the specified new character attributes. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetStyle` (page 99) Deprecated in Mac OS X v10.4

Sets new character attributes, in the specified edit structure, for the current selection range. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEStyleInsert` (page 101) Deprecated in Mac OS X v10.4

Inserts the specified text immediately before the selection range or the insertion point in the edit structure's text and applies the specified character attributes to the text, redrawing the text if necessary. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEUseStyleScrap` (page 105) Deprecated in Mac OS X v10.4

Assigns new character attributes to the specified range of text in the designated edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Customizing TextEdit

`TECustomHook` (page 75) Deprecated in Mac OS X v10.4

Replaces a default TextEdit hook function with a customized function and returns the address of the replaced function. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetDoTextHook` (page 80) Deprecated in Mac OS X v10.4

Obtains a universal procedure pointer to your do-text-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetFindWordHook` (page 80) Deprecated in Mac OS X v10.4

Obtains a universal procedure pointer to your set-find-word-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetRecalcHook` (page 83) Deprecated in Mac OS X v10.4

Obtains a universal procedure pointer to your recalculation callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetClickLoop` (page 95) Deprecated in Mac OS X v10.4

Installs the address of the application-supplied click loop function in the `clikLoop` field of the edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetDoTextHook` (page 96) Deprecated in Mac OS X v10.4

Sets your do-text-hook callback to be used by TextEdit. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetFindWordHook` (page 96) Deprecated in Mac OS X v10.4

Sets your set-find-word-hook callback to be used by TextEdit. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetRecalcHook` (page 97) Deprecated in Mac OS X v10.4

Sets your recalculation callback to be used by TextEdit. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Displaying and Scrolling Text

`TEAutoView` (page 72) Deprecated in Mac OS X v10.4

Enables and disables automatic scrolling of the text in the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TECalText` (page 72) Deprecated in Mac OS X v10.4

Recalculates the beginnings of all lines of text in the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetHeight` (page 81) Deprecated in Mac OS X v10.4

Returns the total height of all of the lines in the text between and including the specified starting and ending lines. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEPinScroll` (page 91) Deprecated in Mac OS X v10.4

Scrolls the text within the view rectangle of the specified edit structure by the designated number of pixels. Scrolling stops when the last line of text is scrolled into view. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEScroll` (page 94) Deprecated in Mac OS X v10.4

Scrolls the text within the view rectangle of the specified edit structure by the designated number of pixels. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESelView` (page 94) Deprecated in Mac OS X v10.4

Ensures, once automatic scrolling has been enabled by a call to the `TEAutoView` function or through the `TEFeatureFlag` function, that the selection range is visible, scrolling it into the view rectangle if necessary. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetAlignment` (page 95) Deprecated in Mac OS X v10.4

Sets the alignment of the specified text in an edit structure so that it is centered, right aligned, or left aligned, or aligned according to the line direction. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TETextBox` (page 103) Deprecated in Mac OS X v10.4

Draws the indicated text in a given rectangle, with the specified alignment. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEUpdate` (page 105) Deprecated in Mac OS X v10.4

Draws the specified text within a given update rectangle. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Initializing TextEdit, Creating an Edit Structure, and Disposing of an Edit Structure

`TEDispose` (page 78) Deprecated in Mac OS X v10.4

Removes a specified edit structure and releases all memory associated with it. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TENew` (page 89) Deprecated in Mac OS X v10.4

Creates and initializes a monostyled edit structure and allocates a handle to it. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEStyleNew` (page 102) Deprecated in Mac OS X v10.4

Creates a multistyled edit structure and allocates a handle to it. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Managing the TextEdit Private Scrap

`TEGetScrapHandle` (page 84) Deprecated in Mac OS X v10.4

Returns a handle to the TextEdit private scrap. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetScrapLength` (page 84) Deprecated in Mac OS X v10.4

Returns the size of the TextEdit private scrap, in bytes. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEScrapHandle` (page 93) Deprecated in Mac OS X v10.4

Returns a handle to the TextEdit private scrap. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetScrapHandle` (page 97) Deprecated in Mac OS X v10.4

Sets a handle to the TextEdit private scrap. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetScrapLength` (page 97) Deprecated in Mac OS X v10.4

Sets the size of the TextEdit private scrap to the specified number of bytes. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Modifying the Text of an Edit Structure

`TECopy` (page 75) Deprecated in Mac OS X v10.4

Copies the text selection range from the edit structure, leaving the selection range intact. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TECut` (page 76) Deprecated in Mac OS X v10.4

Removes the current selection range from the text of the designated edit structure, redrawing the text as necessary. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEDelete` (page 78) Deprecated in Mac OS X v10.4

Removes the selected range of text from the designated edit structure, redrawing the remaining text as necessary. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEFromScrap` (page 80) Deprecated in Mac OS X v10.4

Copies the contents of the desk scrap to the TextEdit private scrap. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEInsert` (page 88) Deprecated in Mac OS X v10.4

Inserts the specified text immediately before the selection range or the insertion point in the text of the designated edit structure, redrawing the text as necessary. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEPaste` (page 91) Deprecated in Mac OS X v10.4

Replaces the edit structure's selected text with the contents of the private scrap and leaves an insertion point after the inserted text. If the selection range is an insertion point, `TEPaste` inserts the contents of the private scrap there. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEStylePaste` (page 103) Deprecated in Mac OS X v10.4

Pastes text and its associated character attribute information from the desk scrap into the edit structure's text at the insertion point—if the current selection range is an insertion point—or it replaces the current selection range. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEToScrap` (page 104) Deprecated in Mac OS X v10.4

Copies the contents of the TextEdit private scrap to the desk scrap. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Setting and Getting an Edit Structure's Text and Character Attribute Information

`TEGetStyleHandle` (page 85) Deprecated in Mac OS X v10.4

Returns the style handle stored in the designated edit structure's `txFont` and `txFace` fields. The style handle points to the associated style structure, not to a copy of it. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetText` (page 86) Deprecated in Mac OS X v10.4

Returns a handle to the text of the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEKey` (page 88) Deprecated in Mac OS X v10.4

Replaces the selection range in the text of the specified edit structure with the input character and positions the insertion point just past the inserted character. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetStyleHandle` (page 100) Deprecated in Mac OS X v10.4

Sets an edit structure's style handle, which is stored in the `txFont` and `txFace` fields. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetText` (page 100) Deprecated in Mac OS X v10.4

Incorporates a copy of the specified text into the designated edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Setting the Caret and Selection Range

`TEClick` (page 73) Deprecated in Mac OS X v10.4

Controls placement and highlighting of the selection range as determined by mouse events. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetHiliteRgn` (page 82) Deprecated in Mac OS X v10.4

Obtains the highlight region for the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEIdle` (page 87) Deprecated in Mac OS X v10.4

When called repeatedly, displays a blinking caret at the insertion point, if any exists, in the text of the specified edit structure of an active window. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TESetSelect` (page 98) Deprecated in Mac OS X v10.4

Sets the selection range (or denotes the insertion point) within the text of the specified edit structure. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Using Byte Offsets and Corresponding Points

`TEGetOffset` (page 82) Deprecated in Mac OS X v10.4

Finds the byte offset of a character in an edit structure's text that corresponds to the specified point. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`TEGetPoint` (page 83) Deprecated in Mac OS X v10.4

Determines the point that corresponds to the specified byte offset of a character and returns the coordinates of that point. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Handling TSM Dialogs

`GetTSMTEDialogDocumentID` (page 57) Deprecated in Mac OS X v10.4

Returns a TSM document ID for the specified dialog. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`GetTSMTEDialogTSMTERecHandle` (page 58) Deprecated in Mac OS X v10.4

Returns a handle to a TSM record for the specified dialog. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`IsTSMTEDialog` (page 65) Deprecated in Mac OS X v10.4

Checks to see if the specified dialog is a TSMTE dialog. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`SetTSMTEDialogDocumentID` (page 70) Deprecated in Mac OS X v10.4

Sets the document ID for the specified dialog. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`SetTSMTEDialogTSMTERecHandle` (page 71) Deprecated in Mac OS X v10.4

Sets a handle to a TSMTE record for the specified dialog. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Working With UPPs for TextEdit Callback Functions

`DisposeCaretHookUPP` (page 53) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a caret-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeDrawHookUPP` (page 53) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a draw-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeEOLHookUPP` (page 53) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to an EOL-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeHighHookUPP` (page 54) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a high-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeHitTestHookUPP` (page 54) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a hit-test hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeNWidthHookUPP` (page 54) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTEClickLoopUPP` (page 55) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a click-loop callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTEDoTextUPP` (page 55) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a do-text callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTEFindWordUPP` (page 55) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a find-word callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTERecalcUPP` (page 56) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a recaluclation callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTextWidthHookUPP` (page 56) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a text-width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTSMTEPostUpdateUPP` (page 56) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a post-update callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeTSMTEPreUpdateUPP` (page 57) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a pre-update callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`DisposeWidthHookUPP` (page 57) Deprecated in Mac OS X v10.4

Disposes of a universal procedure pointer (UPP) to a width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeCaretHookUPP` (page 58) Deprecated in Mac OS X v10.4

Calls a caret-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeDrawHookUPP` (page 59) Deprecated in Mac OS X v10.4

Calls a draw-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeEOLHookUPP` (page 59) Deprecated in Mac OS X v10.4

Calls an EOL-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeHighHookUPP` (page 60) Deprecated in Mac OS X v10.4

Calls a high-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeHitTestHookUPP` (page 60) Deprecated in Mac OS X v10.4

Calls a hit-test hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeNWidthHookUPP` (page 61) Deprecated in Mac OS X v10.4

Calls a width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTEClickLoopUPP` (page 61) Deprecated in Mac OS X v10.4

Calls a click-loop callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTEDoTextUPP` (page 61) Deprecated in Mac OS X v10.4

Calls a do-text callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTEFindWordUPP` (page 62) Deprecated in Mac OS X v10.4

Calls a find-word callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTERecalcUPP` (page 62) Deprecated in Mac OS X v10.4

Calls a recalculation callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTextWidthHookUPP` (page 63) Deprecated in Mac OS X v10.4

Calls a text-width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTSMTEPostUpdateUPP` (page 63) Deprecated in Mac OS X v10.4

Calls a post-update callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeTSMTEPreUpdateUPP` (page 64) Deprecated in Mac OS X v10.4

Calls a pre-update callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`InvokeWidthHookUPP` (page 64) Deprecated in Mac OS X v10.4

Calls a width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewCaretHookUPP` (page 65) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a caret-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewDrawHookUPP` (page 66) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a draw-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewEOLHookUPP` (page 66) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to an EOL-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewHighHookUPP` (page 66) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a high-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewHitTestHookUPP` (page 67) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a hit-test hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewNWidthHookUPP` (page 67) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTEClickLoopUPP` (page 67) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a click-loop callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTEDoTextUPP` (page 68) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a do-text callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTEFindWordUPP` (page 68) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a find-word callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTERecalcUPP` (page 69) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a recalculation callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTextWidthHookUPP` (page 69) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a text-width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTSMTEPostUpdateUPP` (page 69) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a post-update callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewTSMTEPreUpdateUPP` (page 70) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a pre-update callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

`NewWidthHookUPP` (page 70) Deprecated in Mac OS X v10.4

Creates a new universal procedure pointer (UPP) to a width-hook callback. (Deprecated. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

Callbacks

CaretHookProcPtr

Defines a pointer to a caret-hook callback.

```
typedef void (*CaretHookProcPtr) (
    const Rect * r,
    TEPtr pTE
);
```

If you name your function `MyCaretHookProc`, you would declare it like this:

```
void CaretHookProcPtr (
    const Rect * r,
    TEPtr pTE
);
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TextEdit.h`

DrawHookProcPtr

Defines a pointer to a draw-hook callback.

```
typedef void (*DrawHookProcPtr) (
    unsigned short textOffset,
    unsigned short drawLen,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE
);
```

If you name your function `MyDrawHookProc`, you would declare it like this:

```
void DrawHookProcPtr (
    unsigned short textOffset,
    unsigned short drawLen,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

EOLHookProcPtr

Defines a pointer to an EOL-hook callback.

```
typedef Boolean (*EOLHookProcPtr) (
    char theChar,
    TEPtr pTE,
    TEHandle hTE
);
```

If you name your function `MyEOLHookProc`, you would declare it like this:

```
Boolean EOLHookProcPtr (
    char theChar,
    TEPtr pTE,
    TEHandle hTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

HighHookProcPtr

Defines a pointer to a high-hook callback.

```
typedef void (*HighHookProcPtr) (
    const Rect * r,
    TEPtr pTE
);
```

If you name your function `MyHighHookProc`, you would declare it like this:

```
void HighHookProcPtr (
    const Rect * r,
    TEPtr pTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

HitTestHookProcPtr

Defines a pointer to a hit-test hook callback.

```
typedef Boolean (*HitTestHookProcPtr) (
    unsigned short styleRunLen,
    unsigned short styleRunOffset,
    unsigned short slop,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE,
    unsigned short * pixelWidth,
    unsigned short * charOffset,
    Boolean * pixelInChar
);
```

If you name your function `MyHitTestHookProc`, you would declare it like this:

```
Boolean HitTestHookProcPtr (
    unsigned short styleRunLen,
    unsigned short styleRunOffset,
    unsigned short slop,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE,
    unsigned short * pixelWidth,
    unsigned short * charOffset,
    Boolean * pixelInChar
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

NWidthHookProcPtr

Defines a pointer to a width-hook callback.

```
typedef unsigned short (*NWidthHookProcPtr) (
    unsigned short styleRunLen,
    unsigned short styleRunOffset,
    short slop,
    short direction,
    void * textBufferPtr,
    short * lineStart,
    TEPtr pTE,
    TEHandle hTE
);
```

If you name your function `MyNWidthHookProc`, you would declare it like this:

```
unsigned short NWidthHookProcPtr (
    unsigned short styleRunLen,
    unsigned short styleRunOffset,
```

```
    short slop,
    short direction,
    void * textBufferPtr,
    short * lineStart,
    TEPtr pTE,
    TEHandle hTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEClickLoopProcPtr

Defines a pointer to a click-loop callback.

```
typedef Boolean (*TEClickLoopProcPtr) (
    TEPtr pTE
);
```

If you name your function `MyTEClickLoopProc`, you would declare it like this:

```
Boolean TEClickLoopProcPtr (
    TEPtr pTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEDoTextProcPtr

Defines a pointer to a do-text callback.

```
typedef void (*TEDoTextProcPtr) (
    TEPtr pTE,
    unsigned short firstChar,
    unsigned short lastChar,
    short selector,
    GrafPtr * currentGrafPort,
    short * charPosition
);
```

If you name your function `MyTEDoTextProc`, you would declare it like this:

```
void TEDoTextProcPtr (
    TEPtr pTE,
    unsigned short firstChar,
    unsigned short lastChar,
```

```
    short selector,
    GrafPtr * currentGrafPort,
    short * charPosition
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEFindWordProcPtr

Defines a pointer to a find-word callback.

```
typedef void (*TEFindWordProcPtr) (
    unsigned short currentPos,
    short caller,
    TEPtr pTE,
    TEHandle hTE,
    unsigned short * wordStart,
    unsigned short * wordEnd
);
```

If you name your function `MyTEFindWordProc`, you would declare it like this:

```
void TEFindWordProcPtr (
    unsigned short currentPos,
    short caller,
    TEPtr pTE,
    TEHandle hTE,
    unsigned short * wordStart,
    unsigned short * wordEnd
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TERecalcProcPtr

Defines a pointer to a recalculation callback.

```
typedef void (*TERecalcProcPtr) (
    TEPtr pTE,
    unsigned short changeLength,
    unsigned short * lineStart,
    unsigned short * firstChar,
    unsigned short * lastChar
);
```

If you name your function `MyTERecalcProc`, you would declare it like this:

```
void TERecalcProcPtr (
    TEPtr pTE,
    unsigned short changeLength,
    unsigned short * lineStart,
    unsigned short * firstChar,
    unsigned short * lastChar
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TextWidthHookProcPtr

Defines a pointer to a width-hook callback.

```
typedef unsigned short (*TextWidthHookProcPtr) (
    unsigned short textLen,
    unsigned short textOffset,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE
);
```

If you name your function `MyTextWidthHookProc`, you would declare it like this:

```
unsigned short TextWidthHookProcPtr (
    unsigned short textLen,
    unsigned short textOffset,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TSMTEPostUpdateProcPtr

Defines a pointer to a post-update callback.

```
typedef void (*TSMTEPostUpdateProcPtr) (
    TEHandle textH,
    long fixLen,
    long inputAreaStart,
    long inputAreaEnd,
    long pinStart,
    long pinEnd,
    long refCon
);
```

If you name your function `MyTSMTEPostUpdateProc`, you would declare it like this:

```
void TSMTEPostUpdateProcPtr (
    TEHandle textH,
    long fixLen,
    long inputAreaStart,
    long inputAreaEnd,
    long pinStart,
    long pinEnd,
    long refCon
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TSMTE.h`

TSMTEPreUpdateProcPtr

Defines a pointer to a pre-udate callback.

```
typedef void (*TSMTEPreUpdateProcPtr) (
    TEHandle textH,
    long refCon
);
```

If you name your function `MyTSMTEPreUpdateProc`, you would declare it like this:

```
void TSMTEPreUpdateProcPtr (
    TEHandle textH,
    long refCon
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TSMTE.h`

WidthHookProcPtr

Defines a pointer to a width-hook callback.

```
typedef unsigned short (*WidthHookProcPtr) (
    unsigned short textLen,
    unsigned short textOffset,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE
);
```

If you name your function `MyWidthHookProc`, you would declare it like this:

```
unsigned short WidthHookProcPtr (
    unsigned short textLen,
    unsigned short textOffset,
    void * textBufferPtr,
    TEPtr pTE,
    TEHandle hTE
);
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

Data Types

CaretHookUPP

Defines a universal procedure pointer (UPP) to a caret-hook callback.

```
typedef CaretHookProcPtr CaretHookUPP;
```

Discussion
For more information, see the description of the CaretHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

Chars

Defines an array of characters.

```
typedef  char Chars[32001];
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

CharsPtr

Defines a data type for a character pointer.

```
typedef char* CharsPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

CharsHandle

Defines a handle to a character pointer.

```
typedef CharsPtr* CharsHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

DrawHookUPP

Defines a universal procedure pointer (UPP) to a draw-hook callback.

```
typedef DrawHookProcPtr DrawHookUPP;
```

Discussion
For more information, see the description of the DrawHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

EOLHookUPP

Defines a universal procedure pointer (UPP) to an EOL-hook callback.

```
typedef EOLHookProcPtr EOLHookUPP;
```

Discussion
For more information, see the description of the EOLHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

HighHookUPP

Defines a universal procedure pointer (UPP) to a high-hook callback.

```
typedef HighHookProcPtr HighHookUPP;
```

Discussion
For more information, see the description of the HighHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

HitTestHookUPP

Defines a universal procedure pointer (UPP) to a hit-test hook callback.

```
typedef HitTestHookProcPtr HitTestHookUPP;
```

Discussion
For more information, see the description of the HitTestHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

LHHandle

Defines a handle to a line-height table pointer.

```
typedef LHPtr *     LHHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

LHElement

Contains height and ascent information.

```
struct LHElement {
    short lhHeight;
    short lhAscent;
};
typedef struct LHElement LHElement;
typedef LHElement * LHPtr;
```

Fields

`lhHeight`

The line height, in points. This is the maximum value for any individual character attribute in the line.

`lhAscent`

The font ascent, in points; this is the maximum value for any individual character attribute in a line.

Discussion

The line-height table, defined by the `LHTable` data type, provides an array of line heights to hold the vertical spacing information for a given edit structure. It also contains line ascent information. The null style structure, defined by the `NullStRec` data type, contains the null scrap which is used to store character attribute information for a null selection.

The line height table holds vertical spacing information for the text of an edit structure. This table parallels the `lineStarts` array in the edit structure itself. Its length equals the edit structure's `nLines` field plus 1 for a dummy entry at the end, just as the `lineStarts` array ends with a dummy entry that has the same value as the length of the text. The table's contents are recalculated whenever the line starting values are themselves recalculated with the `TECalText` function or whenever an editing action causes recalibration.

The line height table is used only if the `lineHeight` and `fontAscent` fields in the edit structure are negative; positive values in those fields specify fixed vertical spacing, overriding the information in the table. The line height table is of type `LHTable`, which is an array of elements of `LHElement`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TextEdit.h`

LHTable

Defines an array of line-height elements.

```
typedef LHElement LHTable[8001];
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`TextEdit.h`

NullStHandle

Defines a handle to a null scrap record pointer.

```
typedef NullStPtr *      NullStHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

NullStRec

Contains the null scrap.

```
struct NullStRec {
    long teReserved;
    StScrpHandle nullScrap;
};
typedef struct NullStRec NullStRec;
typedef NullStRec * NullStPtr;
```

Fields

`teReserved`
This field is reserved for future expansion.

`nullScrap`
A handle to the style scrap structure.

Discussion
The `NullSTRec` data type defines the null style structure.

The null style structure contains the null scrap, which is used to store the character attribute information for a null selection (insertion point). A number of functions either write this character attribute information to the null scrap or read it from this scrap (to be applied to inserted text). The null scrap is created and initialized when an application calls `TEStyleNew` to create a multistyled edit structure. The null scrap is retained for the life of the edit structure; it is destroyed when `TEDispose` destroys the edit structure and releases the memory allocated for it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

NWidthHookUPP

Defines a universal procedure pointer (UPP) to a width-hook callback.

```
typedef NWidthHookProcPtr NWidthHookUPP;
```

Discussion
For more information, see the description of the NWidthHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

ScrpSTElement

Contains the scrap style table.

```
struct ScrpSTElement {
    long scrpStartChar;
    short scrpHeight;
    short scrpAscent;
    short scrpFont;
    StyleField scrpFace;
    short scrpSize;
    RGBColor scrpColor;
};
typedef struct ScrpSTElement ScrpSTElement;
typedef ScrpSTElement ScrpSTTable[1601];
```

Fields

`scrpStartChar`
: The offset to the beginning of a style structure in the scrap.

`scrpHeight`
: The line height. You can determine the line height and the font ascent using the QuickDraw function `GetFontInfo`.

`scrpAscent`
: The font ascent. See `scrpHeight`.

`scrpFont`
: The font family ID.

`scrpFace`
: The character style (such as plain, bold, underline).

`scrpSize`
: The size, in points.

`scrpColor`
: The RGB (red, green, blue) color for the style scrap.

Discussion
The style scrap structure contains the scrap style table. Unlike the main style table for an edit structure, the scrap style table may contain duplicate elements; the entries in the table correspond one-to-one with the style runs in the text. The `scrpStartChar` field of each entry gives the starting position for the run.

The `scrpStyleTab` data type defines the scrap style table data structure, which is an array of scrap style element structures. The `ScrpSTElement` data type defines each scrap style element structure.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

ScrpSTTable

Contains an array of scrap style elements.

```
typedef ScrpSTElement ScrpSTTable[1601];
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

STElement

Contains one entry for each distinct set of character attributes used in the text of an edit structure.

```
struct STElement {
    short stCount;
    short stHeight;
    short stAscent;
    short stFont;
    StyleField stFace;
    short stSize;
    RGBColor stColor;
};
typedef struct STElement STElement;
typedef STElement * STPtr;
```

Fields

`stCount`
: A reference count of character runs using this set of character attributes.

`stHeight`
: The line height for this run, in points.

`stAscent`
: The font ascent for this run, in points.

`stFont`
: The font family ID.

`stFace`
: The character style (bold, italic, and so forth). This field consists of two bytes. The low-order byte contains the character style. TextEdit uses the high bit (bit 15) of the high-order byte to store the style run direction: it uses 0 for left-to-right text, and 1 for right-to-left text.

`stSize`
: The text size, in points.

`stColor`
: The RGB (red, green, blue) color.

Discussion
The style table contains one entry for each distinct set of character attributes used in the text of an edit structure. Each entry is defined in a style element structure. The size of the table is given by the `nStyles` field of the style structure. There is no duplication; each set of character attributes appears exactly once in the table. A reference count tells how many times each set of attributes is used in the table. The `TEStyleTable` data type defines the style table. Th `eSTElement` data type defines the style element structure.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

STHandle

Defines a handle to a style table pointer.

```
typedef STPtr *     STHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

StScrpHandle

Defines a handle to a scrap style table pointer.

```
typedef StScrpPtr *     StScrpHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

StScrpRec

Contains information used by functions to store character attribute information temporarily.

```
struct StScrpRec {
    short scrpNStyles;
    ScrpSTTable scrpStyleTab;
};
typedef struct StScrpRec StScrpRec;
typedef StScrpRec * StScrpPtr;
```

Fields

`scrpNStyles`
The number of style runs (sets of character attributes) used in the text. This determines the size of the style table. When character attribute information is written to the null scrap, this field is set to 1; when the character attribute information is removed, this field is set to 0.

`scrpStyleTab`
The scrap style table containing an element for each style run (set of character attributes).

Discussion
The style scrap structure, defined by the `StScrpRec` data type, is used by functions to store character attribute information temporarily. The scrap style table, defined by the `scrpStyleTab` data type, is contained in the style scrap structure. The scrap style element structure, defined by the `ScrpSTElement` data type, contains the character attribute information for an element in the scrap style table. One scrap style element structure exists for each sequential attribute change in the associated text.

The style scrap is used for storing character attribute information associated with the current text selection or insertion point, character attribute information to be applied to text, or multistyled text that is cut or copied. When multistyled text is cut or copied, the character attribute information is written to both the style scrap and the desk scrap.

In most cases, the style scrap is created dynamically as needed by functions. However, a style scrap structure can be created directly without using the `TEGetStyleScrapHandle` function; the character attribute information written to it can be applied to inserted text through `TEStyleInsert` or to existing text through `TEUseStyleScrap`.

The format of the style scrap is defined by a style scrap structure of type `STScrpRec`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

StyleRun

Contains information for a style run.

```
struct StyleRun {
    short startChar;
    short styleIndex;
};
typedef struct StyleRun StyleRun;
```

Fields
`startChar`
The starting character position.
`styleIndex`
The run's index in the style table.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEClickLoopUPP

Defines a universal procedure pointer (UPP) to a click-loop callback.

```
typedef TEClickLoopProcPtr TEClickLoopUPP;
```

Discussion
For more information, see the description of the TEClickLoopUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEDoTextUPP

Defines a universal procedure pointer (UPP) to a do-text callback.

```
typedef TEDoTextProcPtr TEDoTextUPP;
```

Discussion
For more information, see the description of the TEDoTextUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEFindWordUPP

Defines a universal procedure pointer (UPP) to a find-word callback.

```
typedef TEFindWordProcPtr TEFindWordUPP;
```

Discussion
For more information, see the description of the TEFindWordUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEHandle

Defines a handle to a TextEdit record pointer.

```
typedef TEPtr* TEHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEIntHook

Defines a data type for a TextEdit integer hook.

```
typedef  short TEIntHook;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEPtr

Defines a pointer to a TextEdit record.

```
typedef TTERec* TEPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TERec

Stores display and editing information for TextEdit.

```
struct TERec {
    Rect destRect;
    Rect viewRect;
    Rect selRect;
    short lineHeight;
    short fontAscent;
    Point selPoint;
    short selStart;
    short selEnd;
    short active;
    WordBreakUPP wordBreak;
    TEClickLoopUPP clickLoop;
    long clickTime;
    short clickLoc;
    long caretTime;
    short caretState;
    short just;
    short teLength;
    Handle hText;
    long hDispatchRec;
    short clikStuff;
    short crOnly;
    short txFont;
    StyleField txFace;
    short txMode;
    short txSize;
    GrafPtr inPort;
    HighHookUPP highHook;
    CaretHookUPP caretHook;
    short nLines;
    short lineStarts[16001];
};
typedef struct TERec TERec;
typedef TERec * TEPtr;
```

Fields

`destRect`

The destination rectangle, in local coordinates.

`viewRect`

The view rectangle, in local coordinates.

`selRect`

The selection rectangle, whose boundaries are defined in local coordinates. This value is the current selection range or insertion point.

`lineHeight`

The vertical spacing of lines of text. Vertical spacing may be fixed or it may vary from line to line, depending upon specific text attributes. If the value of `lineHeight` is greater than 0, this field specifies the fixed vertical distance from the ascent line of one line of text down to the ascent line of the next.

If the value of `lineHeight` is less than 1, then this field specifies the vertical distance from the ascent line of one line of text down to the ascent line of the next calculated independently for each line, based on the maximum value for any individual character attribute on that line.

`fontAscent`

The font ascent line. If the value of `fontAscent` is greater than 0, this field specifies how far above the base line the pen is positioned to begin drawing the caret or highlighting.

For single-spaced text, this is the height of the text in pixels (the height of the tallest characters in the font from the base line). If the value of `fontAscent` is less than 1, this field specifies the font ascent calculated independently for each line, based on maximum value for any individual character attribute on that line.

`selPoint`

The point selected with the mouse, in the local coordinates of the current graphics port. The assembly-language offset for this field is named `teSelPoint`.

`selStart`

The byte offset of the beginning of a selection range. Note that byte offset 0 refers to the first byte in the text buffer.

`selEnd`

The byte offset of the end of a selection range. To include that byte, this value must be 1 greater than the position of the last byte offset of the text.

`active`

This field is used internally by TextEdit. It is set when an edit structure is activated through `TEActivate` and then reset when the edit structure is rendered inactive through `TEDeactivate`. To ensure future compatibility, use `TEActivate` or `TEDeactivate` to access this field.

`wordBreak`

A universal procedure pointer to the structure's word selection break function. This function determines the word that is highlighted when the user double-clicks in the text and the position at which text is wrapped at the end of a line.

`clickLoop`

A universal procedure pointer to the click loop function.The specified click loop function is called repeatedly by the `TEClick` function as long as the mouse button is held down within the text.

`clickTime`

This field is for internal use only.

`clickLoc`

This field is for internal use only.

`caretTime`

This field is for internal use only.

`caretState`

This field is for internal use only.

`just`

The type of text alignment: default (according to primary line direction), left, center, or right.

`teLength`

The number of bytes in the text to be edited. For two-byte systems, potentially twice the number of characters. Initially set to zero. The maximum length is 32767 bytes.

`hText`

A handle to the text. Initially, it points to a zero-length block of text in the heap.

`hDispatchRec`

A handle to the TextEdit dispatch structure. This field is for internal use only; do not modify this field, or copy it to another edit structure. Each edit structure has its own dispatch structure. Attempting to use the dispatch structure of one edit structure with another edit structure can cause TextEdit to crash.

`clikStuff`

This field is for internal use only. TextEdit sets this field to reflect whether the most recent mouse-down event occurred on the leading or trailing edge of a glyph. TextEdit uses this value in determining a caret position.

`crOnly`

A value specifying whether or not text wraps at the right edge of the destination rectangle. If `crOnly` is positive, text does wrap. Otherwise, new lines are displayed only at Carriage Returns.

If `crOnly` is negative, new lines are specified explicitly by `Return` characters only; text does not wrap at the edge of the destination rectangle. (This is useful in an application similar to a programming-language editor, where you may not want a single line of code to be split onto two lines.

`txFont`

The font of all the text in the edit structure, if the `txSize` field of this edit structure 0. If you change this value, the entire text of this edit structure has the new characteristic when it is redrawn also remember to change the `lineHeight` and `fontAscent` fields.

If the `txSize` field is –1, this field combines with `txFace` to hold a handle to the associated style structure.

`txFace`

The character attributes of all the text in an edit structure, if the `txSize` field of this edit structure 0. If you change this value, the entire text of this edit structure has the new characteristic when it is redrawn also, remember to change the `lineHeight` and `fontAscent` fields as well.

If the `txSize` field is –1, this field combines with `txFont` to hold a handle to the associated style structure.

`txMode`

The pen mode of all the text in the edit structure. If you change this value, the entire text of this edit structure has the new characteristic when it is redrawn; also, remember to change the `lineHeight` and `fontAscent` fields as well.

`txSize`

Depending on its value, `txSize` either contains the point size of all of the text or it acts as a flag indicating whether or not there is associated character attribute information. If `txSize` 0, this is a monostyled edit structure, that is, all text is set in a single font, size, and face, and the value of `txSize` is the size of the text. If `txSize` is –1, the edit structure contains associated character attribute information and the `txFont` and `txFace` fields combine to form a handle to the style structure.

`inPort`

A pointer to the graphics port associated with this edit structure.

`highHook`

A pointer to the function that deals with text highlighting. In assembly language, the `highHook` field is located at the offset `teHiHook`.

`caretHook`

A pointer to the function that controls the appearance of the caret. In assembly language, the `caretHook` field is located at the offset `teCarHook`.

`nLines`

The number of lines in the text.

`lineStarts`

An array containing the character position of the first character in each line. It is declared to have 16001 elements to comply with Pascal range checking. This is a dynamic data structure, having only as many elements as needed. TextEdit calculates these values internally, so do not change the elements of the `lineStarts` array. Because this data structure grows and shrinks, the size of the edit structure changes.

Discussion
The edit structure, defined by the `TERec` data type, stores the display and editing information for TextEdit. Along with various subsidiary data structures, the style structure, defined by the `TEStyleRec` data type, stores the character attribute information for the text of the edit structure.

The edit structure contains display, storage, styling, and other information. Although some fields are used differently for multistyled edit structures and monostyled edit structures, the structure of an edit structure is the same whether the text is multistyled or monostyled.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TERecalcUPP

Defines a universal procedure pointer (UPP) to a recalculation callback.

```
typedef TERecalcProcPtr TERecalcUPP;
```

Discussion
For more information, see the description of the TERecalcUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEStyleRec

Stores the character attribute information for the text of a multistyled edit structure.

```
struct TEStyleRec {
    short nRuns;
    short nStyles;
    STHandle styleTab;
    LHHandle lhTab;
    long teRefCon;
    NullStHandle nullStyle;
    StyleRun runs[8001];
};
typedef struct TEStyleRec TEStyleRec;
typedef TEStyleRec * TEStylePtr;
```

Fields

`nRuns`
: The number of style runs in the text.

`nStyles`
: The number of distinct sets of character attributes used in the text; this forms the size of the style table.

`styleTab`
: A handle to the style table.

`lhTab`
: A handle to the line height table.

`teRefCon`
: A reference constant for use by applications. The application can use this 32-bit field to suit its needs.

`nullStyle`
: A handle to the style scrap structure used to store the character attribute information for a null selection.

`runs`
: A table of style runs that is of indefinite length.

Discussion

The style structure stores the character attribute information for the text of a multistyled edit structure. If an edit structure has associated character attribute information, its `txFont` and `txFac` e fields combine to hold a style handle, of type `TEStyleHandle`, to its style structure. The text is divided into style runs, summarized in the style run table, of type `StyleRun`, which is part of the style structure. Each entry in the style run table gives the starting character position of a run and an index into the style table, of type `TEStyleTable`.

The style table element pointed to by the style run index describes the character attributes for that run.

To determine the length of a run, you subtract its start position from that of the next entry in the style run table. A dummy entry at the end of the style run table delimits the length of the last run; its start position is equal to the overall number of characters in the text, plus 1. The `TEStyleRec` data type defines the style structure.

The style run table, defined by the `StyleRun` data type, is an array that contains the boundaries of each style run and an index to its character attribute information in the style element array. The style table, defined by the `TEStyleTable` data type, contains one entry for each distinct set of character attributes used in the text of the edit structure.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEStyleTable

Defines and array of style elements.

```
typedef STElement TEStyleTable[1777];
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TextStyle

Contains text style information.

```
struct TextStyle {
    short tsFont;
    StyleField tsFace;
    short tsSize;
    RGBColor tsColor;
};
typedef struct TextStyle TextStyle;
typedef TextStyle * TextStylePtr;
```

Fields

`tsFont`
The font family number.

`tsFace`
The character style (bold, italic, plain, and so forth).

`tsSize`
The text size in points.

`tsColor`
The RGB (red, green, blue) color.

`tsColor`

Discussion
Text style structures, which are passed as variables or reference parameters, are used for communicating character attribute information between the application and several TextEdit functions, such as `TEContinuousStyle` and `TEReplaceStyle`. They carry the same information as the style element structures in the style table, but without the reference count, line height, and font ascent.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TextWidthHookUPP

Defines a universal procedure pointer (UPP) to a width-hook callback.

```
typedef TextWidthHookProcPtr TextWidthHookUPP;
```

Discussion
For more information, see the description of the TextWidthHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TSMDialogPeek

Defines a data type for a TSM dialog pointer.

```
typedef TSMDialogPtr TSMDialogPeek;
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`TSMTE.h`

TSMDialogPtr

Defines a pointer to a TSM dialog record.

```
typedef TSMDialogRecord* TSMDialogPtr;
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`TSMTE.h`

TSMDialogRecord

Contains information for a TSM dialog record.

```
struct TSMDialogRecord {
  DialogRecord        fDialog;
  TSMDocumentID       fDocID;
  TSMTERecHandle      fTSMTERecH;
  long                fTSMTERsvd[3];
};
typedef struct TSMDialogRecord  TSMDialogRecord;
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
TSMTE.h

TSMTEPostUpdateUPP

Defines a universal procedure pointer (UPP) to a post-update callback.

```
typedef TSMTEPostUpdateProcPtr TSMTEPostUpdateUPP;
```

Discussion
For more information, see the description of the TSMTEPostUpdateUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
TSMTE.h

TSMTEPreUpdateUPP

Defines a universal procedure pointer (UPP) to a pre-update callback.

```
typedef TSMTEPreUpdateProcPtr TSMTEPreUpdateUPP;
```

Discussion
For more information, see the description of the TSMTEPreUpdateUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
TSMTE.h

TSMTERec

Defines a TSMTE record structure.

```
struct TSMTERec {
    TEHandle textH;
    TSMTEPreUpdateUPP preUpdateProc;
    TSMTEPostUpdateUPP postUpdateProc;
    long updateFlag;
    long refCon;
};
typedef struct TSMTERec TSMTERec;
typedef TSMTERec * TSMTERecPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
TSMTE.h

TSMTERecHandle

Defines a handle to a TSMTE record pointer.

```
typedef TSMTERecPtr * TSMTERecHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TSMTE.h`

WidthHookUPP

Defines a universal procedure pointer (UPP) to a width-hook callback.

```
typedef WidthHookProcPtr WidthHookUPP;
```

Discussion
For more information, see the description of the WidthHookUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

Constants

Auto Idling Flag

Enables automatic idling in an event loop.

```
enum {
    teFIdleWithEventLoopTimer = 7
};
```

Auto Scroll Constant

Specifies automatic scrolling

```
enum {
        kTSMTEAutoScroll = 1
};
```

Do Text Selectors

Specify constants for identifying `TEDoTextSelectors`.

```
enum {
    teFind = 0,
    teHighlight = 1,
    teDraw = -1,
    teCaret = -2
};
```

Find Word Identification Constants

Specify constants for identifying the routine that called `FindWord`.

```
enum {
    teWordSelect = 4,
    teWordDrag = 8,
    teFromFind = 12,
    teFromRecal = 16
};
```

Hook Constants

Specify offsets into the `TEDispatchRec` data structure.

```
enum {
    EOLHook = 0,
    DRAWHook = 4,
    WIDTHHook = 8,
    HITTESTHook = 12,
    nWIDTHHook = 24,
    TextWidthHook = 28
};
```

Inline Input Flag

Specifies to use inline input service.

```
enum {
    teFUseTextServices = 4
};
```

Constants

`teFUseTextServices`

Use inline input service. This flag is no longer in use.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

Signature and Interface Constants

Specify a TSM TextEdit signature or interface.

```
enum {
    kTSMTESignature = 'tmTE',
    kTSMTEInterfaceType = 'tmTE'
};
```

Style Mode Constants

Used to set and replace style modes.

```
enum {
    fontBit = 0,
    faceBit = 1,
    sizeBit = 2,
    clrBit = 3,
    addSizeBit = 4,
    toggleBit = 5
};
```

Text Alignment Constants

Specify justification (word alignment) styles.

```
enum {
    teJustLeft = 0,
    teJustCenter = 1,
    teJustRight = -1,
    teForceLeft = -2,
    teFlushDefault = 0,
    teCenter = 1,
    teFlushRight = -1,
    teFlushLeft = -2
};
```

Constants

`teFlushDefault`

Align according to primary line direction

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teCenter`

Centered for all scripts

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teFlushRight`

Right aligned for all scripts

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teFlushLeft`

Left aligned for all scripts

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

Discussion
You can use these constants to specify the text alignment through the `align` parameter of the `TESetAlignment` (page 95) and `TETextBox` (page 103) functions. For compatibility, the previous names of these constants (`teJustLeft` , `teJustCenter` , `teJustRight` , and `teForceLeft`) are still supported.

Text Custom Hook Constants

Specify a selector for a TextEdit hook function.

```
enum {
    intEOLHook = 0,
    intDrawHook = 1,
    intWidthHook = 2,
    intHitTestHook = 3,
    intNWidthHook = 6,
    intTextWidthHook = 7,
    intInlineInputTSMTEPreUpdateHook = 8,
    intInlineInputTSMTEPostUpdateHook = 9
};
```

Constants

`intEOLHook`
End-of-line hook
Available in Mac OS X v10.0 and later.
Declared in `TextEdit.h`.

`intDrawHook`
Draw hook
Available in Mac OS X v10.0 and later.
Declared in `TextEdit.h`.

`intWidthHook`
Width measurement hook
Available in Mac OS X v10.0 and later.
Declared in `TextEdit.h`.

`intHitTestHook`
Hit test hook
Available in Mac OS X v10.0 and later.
Declared in `TextEdit.h`.

`intNWidthHook`
New width measurement hook
Available in Mac OS X v10.0 and later.
Declared in `TextEdit.h`.

`intTextWidthHook`
Text width measurement hook (low-memory global width measurement hook)
Available in Mac OS X v10.0 and later.
Declared in `TextEdit.h`.

`intInlineInputTSMTEPreUpdateHook`
Specifies a `TSMTEPreUpdateProcPtr` callback.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`intInlineInputTSMTEPostUpdateHook`
Specifies a `TSMTEPostUpdateProcPtr` callback.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

Discussion
To specify a default TextEdit hook function with a customized function, you specify one of the following constants as the value of the `which` parameter to the `TECustomHook` (page 75) function.

Text Feature Action Constants

Specify the action to be performed on a feature.

```
enum {
    teBitClear = 0,
    teBitSet = 1,
    teBitTest = -1
};
```

Constants
`teBitClear`
Disables the specified feature

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teBitSet`
Enables the specified feature. If `teBitTest` returns `teBitSet`, the feature is enabled; if it returns `teBitClear`, it is disabled.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teBitTest`
Returns the current setting of the specified feature

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

Discussion
To specify the action to be performed on a feature, you specify one of these constants as the value of the `action` parameter to the `TEFeatureFlag` (page 79) function.

To test for the availability of these features, you can call the `Gestalt` function with the `gestaltTextEditVersion` selector. A result of `gestaltTE4` or greater returned in the response parameter indicates that outline highlighting and text buffering are available. A result of `gestaltTE5` or greater returned in the response parameter indicates that the two inline input features are available.

Version Notes
The inline input features are also available on version 6.0.7 systems with non-Roman script systems installed. However, there is no Gestalt constant that indicates this availability.

Text Feature Constants

Specify feature or bit definitions for the function `TEFeatureFlag`.

```
enum {
    teFAutoScroll = 0,
    teFTextBuffering = 1,
    teFOutlineHilite = 2,
    teFInlineInput = 3,
    teFUseWhiteBackground = 4,
    teFUseInlineInput = 5,
    teFInlineInputAutoScroll = 6
};
```

Constants

`teFAutoScroll`

Automatic scrolling. You can use the `TEFeatureFlag` function to turn automatic scrolling on and off as an alternative to calling `TEAutoView`. The effect is the same.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teFTextBuffering`

Text buffering. The `teFTextBuffering` selector enables or disables text buffering for performance improvements of 2-byte scripts. This is a global buffer, as opposed to the `TEKey` function's internal 2-byte buffer, and it is used across all active edit structures.

Exercise care when you enable the text-buffering capability in more than one active structure; otherwise, the bytes that are buffered from one edit structure may appear in another edit structure.

Ensure that buffering is not turned off in the middle of processing a 2-byte character. To guarantee the integrity of your structure, it is important that you wait for an idle event before you disable buffering or enable buffering in a second edit structure.

When text buffering is enabled, ensure that the `TEIdle` (page 87) function is called before any pause of more than a few ticks—for example, before the Event Manager function `WaitNextEvent`. A possibility of a long delay before characters appear on the screen exists, especially in non-Roman systems. If you do not call `TEIdle`, the characters can end up in the edit structure of another application.

If text buffering is enabled on a non-Roman script system and the keyboard has changed, TextEdit flushes the text of the current script from the buffer before bringing characters of the new script into the buffer.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teFOutlineHilite`

Outline highlighting. The `teFOutlineHilite` selector specifies outline highlighting as the feature for which an action is to be performed. If a highlighted region exists in an edit structure and the window is inactive, then the highlighted region is outlined or framed.

In the case that outline highlighting is enabled and the current selection range is an insertion point, the caret is then drawn in a gray pattern so that it appears dimmed. To do the framing and caret dimming, TextEdit temporarily replaces the current address in the `highHook` and `caretHook` fields of the edit structure, redraws the caret or the highlighted region, and then immediately restores the hooks to their previous addresses.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`teFInlineInput`

Inline input. You must deactivate an edit structure (using `TEDeactivate`) before changing the state of the feature bits or any fields in the edit structure.

In the future, other text services may use this same mechanism. If you follow the guidelines specified here, your application should also work with future text services. When an inline edit session begins, inline input also sets the `teFInlineInput` bit to provide the following features so that inline input works correctly with TextEdit: disabling font and keyboard synchronization, forcing a multiple-line selection to be highlighted line by line using a separate rectangle for each line rather than using a minimum number of rectangles for optimization, and highlighting a line only to the edge of the text rather than beyond the text to the edge of the view rectangle.

The `teFInlineInput` bit is cleared by inline input when an inline session ends. Use the `teFInlineInput` constant in the feature parameter of `TEFeatureFlag` to include these features in your application even when inline input is not installed. Be careful about changing the state of this bit if the `teFUseTextServices` bit is set. Again, the edit structure should always be deactivated before you change the state of the `teFInlineInput` bit. If you clear the `teFUseTextServices` bit and you set the `teFInlineInput` bit, inline input is disabled, but your application retains the features listed above.

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

Discussion
To identify or adjust a feature, you specify one of these constants as the value of the `feature` parameter to the `TEFeatureFlag` (page 79) function.

Text Styling Constants

Specify character attributes.

```
enum {
    doFont = 1,
    doFace = 2,
    doSize = 4,
    doColor = 8,
    doAll = 15,
    addSize = 16,
    doToggle = 32
};
```

Constants
`doFont`

Sets the font family ID

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`doFace`

Sets the character style

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`doSize`
Sets the type size

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`doColor`
Sets the color

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`doAll`
Sets all attributes

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`addSize`
Increases or decreases the current type size

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

`doToggle`
Modifies the mode

Available in Mac OS X v10.0 and later.

Declared in `TextEdit.h`.

Discussion
You can use these constants (singly or in combination) to specify character attributes, through the `mode` parameter of the `TEContinuousStyle` (page 74) , `TESetStyle` (page 99) , and `TEReplaceStyle` (page 92) functions.

Result Codes

In addition to `noErr`, the most common result code returned by TextEdit is listed below.

Result Code	Value	Description
`noScrapErr`	-100	Scrap does not exist (not initalized). Available in Mac OS X v10.0 and later.

Deprecated TextEdit Reference (Not Recommended) Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

DisposeCaretHookUPP

Disposes of a universal procedure pointer (UPP) to a caret-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeCaretHookUPP (
   CaretHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

DisposeDrawHookUPP

Disposes of a universal procedure pointer (UPP) to a draw-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeDrawHookUPP (
   DrawHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

DisposeEOLHookUPP

Disposes of a universal procedure pointer (UPP) to an EOL-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeEOLHookUPP (
   EOLHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

DisposeHighHookUPP

Disposes of a universal procedure pointer (UPP) to a high-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeHighHookUPP (
   HighHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

DisposeHitTestHookUPP

Disposes of a universal procedure pointer (UPP) to a hit-test hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeHitTestHookUPP (
   HitTestHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

DisposeNWidthHookUPP

Disposes of a universal procedure pointer (UPP) to a width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeNWidthHookUPP (
   NWidthHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

DisposeTEClickLoopUPP

Disposes of a universal procedure pointer (UPP) to a click-loop callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTEClickLoopUPP (
   TEClickLoopUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

DisposeTEDoTextUPP

Disposes of a universal procedure pointer (UPP) to a do-text callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTEDoTextUPP (
   TEDoTextUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

DisposeTEFindWordUPP

Disposes of a universal procedure pointer (UPP) to a find-word callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTEFindWordUPP (
   TEFindWordUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

DisposeTERecalcUPP

Disposes of a universal procedure pointer (UPP) to a recaluclation callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTERecalcUPP (
   TERecalcUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

DisposeTextWidthHookUPP

Disposes of a universal procedure pointer (UPP) to a text-width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTextWidthHookUPP (
   TextWidthHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

DisposeTSMTEPostUpdateUPP

Disposes of a universal procedure pointer (UPP) to a post-update callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTSMTEPostUpdateUPP (
   TSMTEPostUpdateUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TSMTE.h`

DisposeTSMTEPreUpdateUPP

Disposes of a universal procedure pointer (UPP) to a pre-update callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeTSMTEPreUpdateUPP (
   TSMTEPreUpdateUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TSMTE.h`

DisposeWidthHookUPP

Disposes of a universal procedure pointer (UPP) to a width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void DisposeWidthHookUPP (
   WidthHookUPP userUPP
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

GetTSMTEDialogDocumentID

Returns a TSM document ID for the specified dialog. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TSMDocumentID GetTSMTEDialogDocumentID (
   DialogRef dialog
);
```

Return Value
See the Text Services Manager documentation for a description of the `TSMDocumentID` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TSMTE.h`

GetTSMTEDialogTSMTERecHandle

Returns a handle to a TSM record for the specified dialog. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TSMTERecHandle GetTSMTEDialogTSMTERecHandle (
   DialogRef dialog
);
```

Return Value
See the description of the `TSMTERecHandle` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TSMTE.h`

InvokeCaretHookUPP

Calls a caret-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeCaretHookUPP (
   const Rect *r,
   TEPtr pTE,
   CaretHookUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeDrawHookUPP

Calls a draw-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeDrawHookUPP (
   unsigned short textOffset,
   unsigned short drawLen,
   void *textBufferPtr,
   TEPtr pTE,
   TEHandle hTE,
   DrawHookUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeEOLHookUPP

Calls an EOL-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Boolean InvokeEOLHookUPP (
   char theChar,
   TEPtr pTE,
   TEHandle hTE,
   EOLHookUPP userUPP
);
```

Return Value
See the Mac Types documentation for a description of the `Boolean` data type.

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeHighHookUPP

Calls a high-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeHighHookUPP (
   const Rect *r,
   TEPtr pTE,
   HighHookUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeHitTestHookUPP

Calls a hit-test hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Boolean InvokeHitTestHookUPP (
   unsigned short styleRunLen,
   unsigned short styleRunOffset,
   unsigned short slop,
   void *textBufferPtr,
   TEPtr pTE,
   TEHandle hTE,
   unsigned short *pixelWidth,
   unsigned short *charOffset,
   Boolean *pixelInChar,
   HitTestHookUPP userUPP
);
```

Return Value
See the Mac Types documentation for a description of the `Boolean` data type.

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeNWidthHookUPP

Calls a width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
unsigned short InvokeNWidthHookUPP (
   unsigned short styleRunLen,
   unsigned short styleRunOffset,
   short slop,
   short direction,
   void *textBufferPtr,
   short *lineStart,
   TEPtr pTE,
   TEHandle hTE,
   NWidthHookUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeTEClickLoopUPP

Calls a click-loop callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Boolean InvokeTEClickLoopUPP (
   TEPtr pTE,
   TEClickLoopUPP userUPP
);
```

Return Value
See the Mac Types documentation for a description of the `Boolean` data type.

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeTEDoTextUPP

Calls a do-text callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeTEDoTextUPP (
   TEPtr pTE,
   unsigned short firstChar,
   unsigned short lastChar,
   short selector,
   GrafPtr *currentGrafPort,
   short *charPosition,
   TEDoTextUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeTEFindWordUPP

Calls a find-word callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeTEFindWordUPP (
   unsigned short currentPos,
   short caller,
   TEPtr pTE,
   TEHandle hTE,
   unsigned short *wordStart,
   unsigned short *wordEnd,
   TEFindWordUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeTERecalcUPP

Calls a recalculation callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeTERecalcUPP (
   TEPtr pTE,
   unsigned short changeLength,
   unsigned short *lineStart,
   unsigned short *firstChar,
   unsigned short *lastChar,
   TERecalcUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeTextWidthHookUPP

Calls a text-width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
unsigned short InvokeTextWidthHookUPP (
   unsigned short textLen,
   unsigned short textOffset,
   void *textBufferPtr,
   TEPtr pTE,
   TEHandle hTE,
   TextWidthHookUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

InvokeTSMTEPostUpdateUPP

Calls a post-update callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeTSMTEPostUpdateUPP (
   TEHandle textH,
   long fixLen,
   long inputAreaStart,
   long inputAreaEnd,
   long pinStart,
   long pinEnd,
   long refCon,
   TSMTEPostUpdateUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TSMTE.h`

InvokeTSMTEPreUpdateUPP

Calls a pre-update callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void InvokeTSMTEPreUpdateUPP (
   TEHandle textH,
   long refCon,
   TSMTEPreUpdateUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TSMTE.h`

InvokeWidthHookUPP

Calls a width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
unsigned short InvokeWidthHookUPP (
   unsigned short textLen,
   unsigned short textOffset,
   void *textBufferPtr,
   TEPtr pTE,
   TEHandle hTE,
   WidthHookUPP userUPP
);
```

Discussion
You should not need to use this function, as the system invokes your callback for you.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

IsTSMTEDialog

Checks to see if the specified dialog is a TSMTE dialog. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Boolean IsTSMTEDialog (
   DialogRef dialog
);
```

Return Value
See the Mac Types documentation for a description of the `Boolean` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TSMTE.h`

NewCaretHookUPP

Creates a new universal procedure pointer (UPP) to a caret-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
CaretHookUPP NewCaretHookUPP (
   CaretHookProcPtr userRoutine
);
```

Return Value
See the description of the `CaretHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewDrawHookUPP

Creates a new universal procedure pointer (UPP) to a draw-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
DrawHookUPP NewDrawHookUPP (
   DrawHookProcPtr userRoutine
);
```

Return Value
See the description of the `DrawHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewEOLHookUPP

Creates a new universal procedure pointer (UPP) to an EOL-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
EOLHookUPP NewEOLHookUPP (
   EOLHookProcPtr userRoutine
);
```

Return Value
See the description of the `EOLHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewHighHookUPP

Creates a new universal procedure pointer (UPP) to a high-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
HighHookUPP NewHighHookUPP (
   HighHookProcPtr userRoutine
);
```

Return Value
See the description of the `HighHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewHitTestHookUPP

Creates a new universal procedure pointer (UPP) to a hit-test hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
HitTestHookUPP NewHitTestHookUPP (
   HitTestHookProcPtr userRoutine
);
```

Return Value
See the description of the `HitTestHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewNWidthHookUPP

Creates a new universal procedure pointer (UPP) to a width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
NWidthHookUPP NewNWidthHookUPP (
   NWidthHookProcPtr userRoutine
);
```

Return Value
See the description of the `NWidthHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewTEClickLoopUPP

Creates a new universal procedure pointer (UPP) to a click-loop callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEClickLoopUPP NewTEClickLoopUPP (
   TEClickLoopProcPtr userRoutine
);
```

Return Value
See the description of the `TEClickLoopUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewTEDoTextUPP

Creates a new universal procedure pointer (UPP) to a do-text callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEDoTextUPP NewTEDoTextUPP (
   TEDoTextProcPtr userRoutine
);
```

Return Value
See the description of the `TEDoTextUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewTEFindWordUPP

Creates a new universal procedure pointer (UPP) to a find-word callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEFindWordUPP NewTEFindWordUPP (
   TEFindWordProcPtr userRoutine
);
```

Return Value
See `TEFindWordUPP` (page 34) for a description of the `TEFindWordUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewTERecalcUPP

Creates a new universal procedure pointer (UPP) to a recalculation callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TERecalcUPP NewTERecalcUPP (
   TERecalcProcPtr userRoutine
);
```

Return Value
See the description of the `TERecalcUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewTextWidthHookUPP

Creates a new universal procedure pointer (UPP) to a text-width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TextWidthHookUPP NewTextWidthHookUPP (
   TextWidthHookProcPtr userRoutine
);
```

Return Value
See the description of the `TextWidthHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`TextEdit.h`

NewTSMTEPostUpdateUPP

Creates a new universal procedure pointer (UPP) to a post-update callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TSMTEPostUpdateUPP NewTSMTEPostUpdateUPP (
   TSMTEPostUpdateProcPtr userRoutine
);
```

Return Value
See `TSMTEPostUpdateUPP` (page 43) for a description of the `TSMTEPostUpdateUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TSMTE.h

NewTSMTEPreUpdateUPP

Creates a new universal procedure pointer (UPP) to a pre-update callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TSMTEPreUpdateUPP NewTSMTEPreUpdateUPP (
   TSMTEPreUpdateProcPtr userRoutine
);
```

Return Value
See the description of the `TSMTEPreUpdateUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TSMTE.h

NewWidthHookUPP

Creates a new universal procedure pointer (UPP) to a width-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
WidthHookUPP NewWidthHookUPP (
   WidthHookProcPtr userRoutine
);
```

Return Value
See the description of the `WidthHookUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
TextEdit.h

SetTSMTEDialogDocumentID

Sets the document ID for the specified dialog. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void SetTSMTEDialogDocumentID (
   DialogRef dialog,
   TSMDocumentID documentID
);
```

Availability
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TSMTE.h`

SetTSMTEDialogTSMTERecHandle

Sets a handle to a TSMTE record for the specified dialog. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void SetTSMTEDialogTSMTERecHandle (
   DialogRef dialog,
   TSMTERecHandle tsmteRecHandle
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TSMTE.h`

TEActivate

Activates the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEActivate (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the specified edit structure.

Discussion
When your application receives notification of an activate event, it can call the `TEActivate` function, which activates an edit structure and highlights the selection range. If the selection range is an insertion point, `TEActivate` simply displays a caret there. Call this function every time the Event Manager function `WaitNextEvent` reports that the window containing the edit structure has become active.

If you do not call `TEActivate` before you call `TEClick`, `TEIdle`, or `TESetSelect`, the selection range is not highlighted, or, if the selection range is set to an insertion point, a caret is not displayed at the insertion point. However, if you have turned on outline highlighting through the `TEFeatureFlag` function for the edit structure, the text of the selection range is framed or a dimmed or an unblinking caret is displayed at the insertion point.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TEAutoView

Enables and disables automatic scrolling of the text in the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEAutoView (
   Boolean fAuto,
   TEHandle hTE
);
```

Parameters

fAuto

A flag indicating whether to enable or disable automatic scrolling. A value of TRUE enables automatic scrolling; a value of FALSE disables automatic scrolling.

hTE

A handle to the edit structure for which automatic scrolling is to be enabled or disabled.

Discussion
The TEAutoView function does not actually scroll the text automatically: TESelView does. However, when fAuto is set to FALSE , a call to TESelView has no effect.

If there is a scroll bar associated with the edit structure, your application must manage scrolling of it. You can replace the default click loop function, which scrolls the text only, with a customized version that also updates the scroll bar.

You can also enable or disable automatic scrolling for an edit structure through the teFAutoScroll feature of the TEFeatureFlag function. For more information, see TEFeatureFlag (page 79).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TECalText

Recalculates the beginnings of all lines of text in the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TECalText (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure whose text lines are to be recalculated.

Discussion
The `TECalText` function updates elements of the `lineStarts` array in an edit structure. Call `TECalText` if you've changed the destination rectangle, the `hText` field, or any other property of the edit structure that pertains to line breaks and the number of characters per line—for example, font, size, style, and so on.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEClick

Controls placement and highlighting of the selection range as determined by mouse events. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEClick (
   Point pt,
   Boolean fExtend,
   TEHandle h
);
```

Parameters

pt

The mouse location in local coordinates at the time the mouse button was pressed, obtainable from the event structure (in global coordinates).

fExtend

A flag denoting the state of the Shift key at the time of the click as indicated by the Event Manager. If the Shift key was held down at the time of the click to extend the selection, pass a value of `TRUE`.

h

A handle to the edit structure whose text is displayed in the view rectangle where the click occurred.

Discussion
Call `TEClick` whenever a mouse-down event occurs in the view rectangle of the edit structure and the window associated with that edit structure is active. The `TEClick` function keeps control until the mouse button is released. Use the QuickDraw function `GlobalToLocal` to convert the global coordinates of the mouse location given in the event structure to the local coordinate system for `pt`.

The `TEClick` function removes highlighting of the old selection range unless the selection range is being extended. If the mouse moves, meaning that a drag is occurring, `TEClick` expands or shortens the selection range accordingly a character at a time. In the case of a double-click, the word where the cursor is positioned becomes the selection range.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEContinuousStyle

Determines whether a given character attribute is continuous over the current selection range. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Boolean TEContinuousStyle (
   short *mode,
   TextStyle *aStyle,
   TEHandle hTE
);
```

Parameters

mode

On input, a pointer to a selector specifying the attributes to be checked. On output, `mode` identifies only those attributes determined to be continuous over the selection range. Possible values for the `mode` parameter are defined in "Text Styling Constants" (page 50).

aStyle

On input, a pointer to a text style structure. On output, this structure contains the values for the `mode` attributes determined to be continuous over the selection.

hTE

A handle to the edit structure containing the selected text whose attributes are to be checked. If the value of `hTE` is a handle to a monostyled edit structure, `TEContinuousStyle` returns the set of character attributes that are consistent for the entire structure.

Return Value

`TRUE` if all of the attributes to be checked are continuous; `FALSE` if none or some are continuous. See the Mac Types documentation for a description of the `Boolean` data type.

Discussion

This function does not modify the text selection. If the current selection range is an insertion point, `TEContinuousStyle` first checks the null scrap. If the null scrap contains character attributes, then they are used based on the value of the `mode` parameter. Otherwise, if the null scrap is empty, `TEContinuousStyle` returns the attributes of the character preceding the insertion point. The `TEContinuousStyle` function always returns `TRUE` in this case, and each field of the text style structure is set if the corresponding bit in the `mode` parameter is set.

Note that fields in the text style structure specified by `aStyle` are only valid if the corresponding bits are set in the `mode` variable.

How the `tsFace` field of the `aStyle` structure is used requires some consideration. For example, if `TEContinuousStyle` returns a `mode` parameter that contains `doFace` and the text style structure `tsFace` field is bold, it means that the selected text is all bold, but may contain other text styles, such as italic, as well. Italic does not apply to all of the selected text, or it would have been included in the `tsFace` field. If the `tsFace` field is an empty set, then all of the selected text is plain.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TECopy

Copies the text selection range from the edit structure, leaving the selection range intact. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TECopy (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure containing the text to be copied.

Discussion

The `TECopy` function copies the text to the private scrap. For text of a monostyled edit structure, the text is written to the private scrap only. For text of a multistyled edit structure, the text is written to the TextEdit private scrap, the character attribute information is written to the TextEdit style scrap, and both are written to the Scrap Manager's desk scrap. Anything previously in the private scrap is deleted before the copied text is written to it.

For both multistyled and monostyled text, if the selection range is an insertion point, `TECopy` empties the TextEdit private scrap. When the selection range is an insertion point and the text is multistyled, `TECopy` has no effect on the null scrap, the style scrap, or the Scrap Manager's desk scrap.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TECustomHook

Replaces a default TextEdit hook function with a customized function and returns the address of the replaced function. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TECustomHook (
   TEIntHook which,
   UniversalProcPtr *addr,
   TEHandle hTE
);
```

Parameters

which

The hook whose default function is to be replaced.

addr

On input, the address of your customized function.

On output, the `addr` parameter contains the address of the function that was previously installed in the field identified by the `which` parameter. This address is returned so that you can daisy-chain functions.

hTE

A handle to the edit structure to be modified.

Discussion

The `TECustomHook` function lets you alter the behavior of TextEdit to better suit your application's requirements and those of the script systems installed. If you replace a default hook function with a customized version that you write in a high-level language, such as Pascal or C, you need to provide assembly-language glue code that utilizes the registers for your high-level language function.

The end-of-line hook, width measurement hook, new width measurement hook, text width measurement hook, draw hook, and hit test hook fields are hook fields in the TextEdit dispatch structure. The `which` parameter identifies the hook whose default function is to be replaced. You use the constants described in "Text Custom Hook Constants" (page 47) to specify a value for this parameter.

Certain precautions are critical in replacing default functions. Before placing the address of your function in the TextEdit dispatch structure, strip the addresses, using the Operating System Utilities `StripAddress` function, to guarantee that your application is 32-bit clean.

Before replacing a TextEdit function with a customized one, determine whether more than one script system is installed, and if so, ensure that your customized function accommodates all of the installed script systems. This avoids the problem of your customized function producing results that are incompatible with the Script Manager.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TECut

Removes the current selection range from the text of the designated edit structure, redrawing the text as necessary. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TECut (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure containing the text to be cut.

Discussion

For monostyled text, the `TECut` function writes the cut text to the private scrap.

For multistyled text, `TECut` writes the cut text to the private scrap and its character attributes to the style scrap it also writes both to the Scrap Manager's desk scrap. For multistyled text, the `TECut` function removes the character attributes from the style structure's style table when the text is cut.

For both monostyled and multistyled text, if the selection range is an insertion point, TextEdit deletes everything from the private scrap. When the selection range is an insertion point and the text is multistyled, `TECut` has no effect on the style scrap or the Scrap Manager's desk scrap.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEDeactivate

Deactivates the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEDeactivate (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the specified edit structure.

Discussion
When the activate event flag is set to deactivate the window, your application can call the `TEDeactivate` function, which changes an edit structure's status from active to inactive and removes the selection range highlighting or the caret.

However, if you turned on outline highlighting through the `TEFeatureFlag` function for the edit structure, the text of the selection range is framed or a dimmed or an unblinking caret is displayed at the insertion point when the structure is deactivated.

Call this function every time the Event Manager function `WaitNextEvent` reports that the window containing the edit structure has become inactive.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEDelete

Removes the selected range of text from the designated edit structure, redrawing the remaining text as necessary. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEDelete (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure containing the text to be deleted.

Discussion

When the `TEDelete` function deletes a selected range of text, it does not transfer the text to either the private scrap or the Scrap Manager's desk scrap.

For multistyled structures, when you use `TEDelete` to delete a selected range of text, the associated character attributes are saved in the null scrap to be applied to characters entered after the text is deleted. When the user clicks in some other area of the text, the character attributes are removed from the null scrap. You can use `TEDelete` to implement the Clear command. The `TEDelete` function recalculates line starts and line heights.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEDispose

Removes a specified edit structure and releases all memory associated with it. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEDispose (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure for which the allocated memory should be released.

Discussion

Call the `TEDispose` function only when you're completely through with an edit structure.

Note that if your program retains a handle to text associated with the edit structure that you are destroying with `TEDispose` , the handle becomes invalid because the `TEDispose` function disposes of it, as well as the dispatch structure handle. If the structure is multistyled, `TEDispose` also disposes all of the style-related handles: `STHandle`, `LHHandle`, `STScrpHandle`, `nullSTHandle`, and `TEStyleHandle`.

To continue to refer to the text after you've destroyed the edit structure, you need to make a copy of the handle in the `hText` field of the edit structure using the Operating System Utilities `HandToHand` function before you call `TEDispose`.

In addition to disposing of the edit structure, the edit structure handle, and the dispatch structure handle, the `TEDispose` function destroys the null scrap associated with the edit structure and releases the memory used for it.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEFeatureFlag

Turns a specified feature on or off or returns the current status of that feature. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
short TEFeatureFlag (
   short feature,
   short action,
   TEHandle hTE
);
```

Parameters

feature
The feature for which the action is to be performed. See "Text Feature Constants" (page 49) for a description of the available values.

action
A selector stipulating that the feature, specified by the `feature` parameter, is to be turned on or off, or that the current status of the feature is to be returned. See "Text Feature Action Constants" (page 48) for a description of the available values.

hTE
A handle to the edit structure for which the action should be performed.

Return Value
The status of the specified feature (if the selector is set to `teBitTest`).

Discussion
You can use the `TEFeatureFlag` function to check the status of additional TextEdit features—automatic scrolling, outline highlighting, and text buffering—and to enable or disable the feature. You can also use this function to disable inline input in a particular edit structure and to enable several features that have been provided so that inline input works correctly with TextEdit.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TEFromScrap

Copies the contents of the desk scrap to the TextEdit private scrap. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
OSErr TEFromScrap (
   void
);
```

Return Value
A result code. See "TextEdit Result Codes" (page 51).

Discussion
You use this function to move monostyled text across applications or between an application and a desk accessory.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TEGetDoTextHook

Obtains a universal procedure pointer to your do-text-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEDoTextUPP TEGetDoTextHook (
   void
);
```

Return Value
See the description of the `TEDoTextUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TEGetFindWordHook

Obtains a universal procedure pointer to your set-find-word-hook callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEFindWordUPP TEGetFindWordHook (
   void
);
```

Return Value
See the description of the `TEFindWordUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetHeight

Returns the total height of all of the lines in the text between and including the specified starting and ending lines. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
long TEGetHeight (
   long endLine,
   long startLine,
   TEHandle hTE
);
```

Parameters

endLine
The number of the last line of text whose height is to be included in the total height. You can specify a value that is greater than or equal to 1 for this parameter.

startLine
The number of the first line of text whose height is to be included in the total height. You can specify a value that is greater than or equal to 1 for this parameter.

hTE
A handle to the edit structure containing the lines of text whose height is to be returned.

Return Value
The total height of all of the designated text lines.

Discussion
For monostyled text, the `TEGetHeight` function uses the value of the edit structure's `lineHeight` field. For multistyled text, it uses the line height element (`LHElement`) of the line height table (`LHTable`). Note that `TEGetHeight` does not take into account the height of any blank lines at the end of the text. You need to consider this when scrolling text.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetHiliteRgn

Obtains the highlight region for the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
OSErr TEGetHiliteRgn (
   RgnHandle region,
   TEHandle hTE
);
```

Return Value
A result code. See "TextEdit Result Codes" (page 51).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetOffset

Finds the byte offset of a character in an edit structure's text that corresponds to the specified point. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
short TEGetOffset (
   Point pt,
   TEHandle hTE
);
```

Parameters

pt
A point in the displayed text of the specified edit structure.

hTE
A handle to the edit structure containing the text.

Return Value
The byte offset of the character at the specified point. In the case of a 2-byte character, the function returns the byte offset of the first byte.

Discussion
The `TEGetOffset` function works for both monostyled and multistyled edit structures.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetPoint

Determines the point that corresponds to the specified byte offset of a character and returns the coordinates of that point. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Point TEGetPoint (
   short offset,
   TEHandle hTE
);
```

Parameters

offset

A byte offset into the text buffer of an edit structure.

hTE

A handle to the edit structure containing the text.

Return Value

The coordinates of the point that corresponds to the specified byte offset. The `TEGetPoint` function returns a valid result even when the edit structure does not contain any text. The point returned is based on the values in the structure's destination rectangle.

In the case of an offset being equal to a line end, which is also the start of the next line, `TEGetPoint` returns a point corresponding to the line start of the next line. In the case of a dual caret, the primary caret position, the one corresponding to the primary line direction, is returned.

See the Mac Types documentation for a description of the `Point` data type.

Discussion

The line height, taken either from the `lineHeight` field for a monostyled edit structure or from the line-height array, `LHElement` , for a multistyled edit structure, is also used to determine the vertical component. Both the text direction and the primary line direction are used to determine the horizontal component.

The `TEGetPoint` function works for both monostyled and multistyled edit structures.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEGetRecalcHook

Obtains a universal procedure pointer to your recalculation callback. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TERecalcUPP TEGetRecalcHook (
   void
);
```

Return Value

See the description of the `TERecalcUPP` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetScrapHandle

Returns a handle to the TextEdit private scrap. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Handle TEGetScrapHandle (
   void
);
```

Return Value
See the Mac Types documentation for a description of the `Handle` data type.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetScrapLength

Returns the size of the TextEdit private scrap, in bytes. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
long TEGetScrapLength (
   void
);
```

Return Value
The size of the TextEdit private scrap, in bytes.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetStyle

Gets character attributes for the specified text. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEGetStyle (
   short offset,
   TextStyle *theStyle,
   short *lineHeight,
   short *fontAscent,
   TEHandle hTE
);
```

Parameters

offset

The offset to the text whose character attributes you want to obtain.

theStyle

On output, points to a structure of type TextStyle that contains the character attributes for the current selection range.

lineHeight

A pointer to a value that specifies the line height.

fontAscent

A pointer to a value that specifies the font ascent.

hTE

A handle to the multistyled edit structure containing the text whose character attributes you want to obtain.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEGetStyleHandle

Returns the style handle stored in the designated edit structure's `txFont` and `txFace` fields. The style handle points to the associated style structure, not to a copy of it. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEStyleHandle TEGetStyleHandle (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the multistyled edit structure containing the style handle to be returned.

Return Value

A handle to the style structure contained in the specified edit structure (of type `TEStyleRec`). Because only multistyled edit structures have style structures, `TEGetStyleHandle` returns `NULL` when used with a monostyled edit structure. See the description of the `TEStyleHandle` data type.

Discussion
To ensure future compatibility, your application should always use this function rather than manipulate the fields of the edit structure directly.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetStyleScrapHandle

Creates a style scrap structure, copies the character attributes associated with the current selection range into it, and returns a handle to it. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
StScrpHandle TEGetStyleScrapHandle (
   TEHandle hTE
);
```

Parameters

hTE

The handle to the edit structure containing the text selection range whose character attributes are to be copied.

Return Value
A handle to the created style scrap structure, or `NULL` if called with a handle to a monostyled structure. See the description of the `StScrpHandle` data type.

Discussion
The `TEGetStyleScrapHandle` function creates a style scrap structure of type `StScrpRec` and copies the character attributes associated with the current selection range of the designated edit structure into it. If the current selection range is an insertion point, `TEGetStyleScrapHandle` first checks the null scrap. If the null scrap contains character attributes, they are written to the newly created style scrap structure. If the null scrap is empty, the attributes associated with the character preceding the insertion point are copied to the style scrap structure.

The `TEGetStyleScrapHandle` function has no impact on the Scrap Manager's desk scrap.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEGetText

Returns a handle to the text of the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
CharsHandle TEGetText (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure containing the text whose handle you want returned. You pass this handle as an input parameter.

Return Value

A handle to the text contained in the specified edit structure. See page 82 for a description of the `CharsHandle` data type.

Discussion

Given an edit structure that contains text, you can use the `TEGetText` function to get a handle to the text itself. The `TEGetText` function doesn't make a copy of the text. Rather, it returns the handle to the text which is stored as a packed array of characters. (This handle belongs to TextEdit your application must not destroy it.) The `teLength` field of the edit structure contains the length of the text whose handle is returned.

The handle of type `CharsHandle` that is returned by `TEGetText` corresponds to the `hText` field of the `TERec` (page 35) structure.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEIdle

When called repeatedly, displays a blinking caret at the insertion point, if any exists, in the text of the specified edit structure of an active window. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEIdle (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure.

Discussion

You need to call `TEIdle` only when the window containing the text is active; the caret is blinked only then. TextEdit observes a minimum blink interval, initially set to 32 ticks. No matter how often you call `TEIdle`, the time between blinks is never less than the minimum interval. (The user can adjust the minimum interval setting with the General Controls control panel.)

To maintain a constant frequency of blinking, you need to call `TEIdle` at least once each time through your main event loop. Call it more than once if your application does an unusually large amount of processing each time through the loop.

Call the Event Manager's `GetCaretTime` function to get the blink rate.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEInsert

Inserts the specified text immediately before the selection range or the insertion point in the text of the designated edit structure, redrawing the text as necessary. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEInsert (
   const void *text,
   long length,
   TEHandle hTE
);
```

Parameters

text
A pointer to the text to be inserted.

length
The number of characters to be inserted.

hTE
A handle to the edit structure containing the text buffer into which the new text is to be inserted.

Discussion
When you call the `TEInsert` function and a range of text is selected, `TEInsert` doesn't affect the selection range. The `TEInsert` function does not check for a 32 KB limit, so your application must ensure that the inserted text does not exceed this text size limit of 32 KB. The `TEInsert` function recalculates line starts and line heights to adjust for the inserted text.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEKey

Replaces the selection range in the text of the specified edit structure with the input character and positions the insertion point just past the inserted character. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEKey (
   CharParameter key,
   TEHandle hTE
);
```

Parameters

key

The input character.

hTE

A handle to the edit structure in whose text the character is to be entered.

Discussion

The TextEdit function `TEKey` allows you to handle key-down events and enter text input through the keyboard. If the selection range is an insertion point, `TEKey` inserts the character. (Two-byte characters are passed one byte at a time.)

If the `key` parameter contains a backspace character, the selection range or the character immediately before the insertion point is deleted. When the primary line direction is right-to-left, the character to the right of the insertion point is deleted. When the primary line direction is left-to-right, the character to the left of the insertion point is deleted.

When the user deletes text up to the beginning of a set of character attributes, `TEKey` saves the attributes in the null scrap's style scrap structure. The attributes are saved temporarily to be applied to characters inserted after the deletion. As soon as the user clicks in another area of the text, `TEKey` removes the attributes. `TEKey` redraws the text as necessary.

Call `TEKey` every time the Event Manager function `WaitNextEvent` reports a keyboard event that your application determines should be handled by TextEdit.

Because `TEKey` inserts every character passed in the `key` parameter, your application must filter all characters which aren't actual text, such as keys typed in conjunction with the Command key.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TENew

Creates and initializes a monostyled edit structure and allocates a handle to it. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEHandle TENew (
   const Rect *destRect,
   const Rect *viewRect
);
```

Parameters

destRect

A pointer to the destination rectangle for the new edit structure, specified in the local coordinates of the current graphics port. This is the area in which text is laid out.

viewRect

A pointer to the view, or visible, rectangle for the new edit structure, specified in the local coordinates of the current graphics port. This is the area of the window in which text is actually displayed.

Return Value

A handle to the newly created edit structure. Your application needs to store the handle to the edit structure that is returned; many functions require it as an input parameter. See the description of the `TEHandle` data type.

Discussion

A monostyled edit structure is one in which all text is restricted to a single font, size, and style. Use the `TENew` function when the text is to be rendered in attributes that are consistent from character to character. Otherwise, use the `TEStyleNew` (page 102) function.

Call `TENew` once for every edit structure you want allocated. Your application should store the handle to the edit structure that is returned; many functions require it as an input parameter. The edit structure assumes the drawing environment of the graphics port.

If your application contains more than one window where text editing occurs, you need to create an edit structure for each window.

Before this function is called, the window must be in the current graphics port.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TENumStyles

Returns the number of character attribute changes contained in the specified range, counting one for the start of the range. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
long TENumStyles (
   long rangeStart,
   long rangeEnd,
   TEHandle hTE
);
```

Parameters

rangeStart

The beginning of the range of text for which the number of style runs (sets of character attributes) or changes is counted and returned.

rangeEnd

The end of the range of text for which the number of style runs (sets of character attributes) or changes is counted and returned.

hTE

A handle to the edit structure containing the range of text.

Return Value
The number of character attribute changes contained in the specified range. This does not necessarily represent the number of unique sets of attributes for the range, because some sets of attributes may be repeated. For monostyled edit structures, `TENumStyles` always returns 1.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEPaste

Replaces the edit structure's selected text with the contents of the private scrap and leaves an insertion point after the inserted text. If the selection range is an insertion point, `TEPaste` inserts the contents of the private scrap there. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEPaste (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure into which the text is to be pasted.

Discussion
When you call `TEPaste`, after it pastes the text from the private scrap, it redraws all of the text as necessary. If the private scrap is empty, `TEPaste` deletes the selection range. If you call `TEPaste` for a multistyled edit structure, it pastes only the text in the private scrap. In this case, `TEPaste` ignores any associated character attribute information stored in the style scrap; instead, it applies the character attributes of the first character of the selection range being replaced to the text. If the selection range is an insertion point, `TEPaste` applies the character attributes of the character preceding the insertion point.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEPinScroll

Scrolls the text within the view rectangle of the specified edit structure by the designated number of pixels. Scrolling stops when the last line of text is scrolled into view. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEPinScroll (
   short dh,
   short dv,
   TEHandle hTE
);
```

Parameters

dh

The distance in pixels that the text is to be scrolled horizontally. A positive value moves the text to the right; a negative value moves the text to the left.

dv

The distance in pixels that the text is to be scrolled vertically. A positive value moves the text down; a negative value moves the text up.

hTE

A handle to the edit structure whose text is to be scrolled.

Discussion

The `TEPinScroll` function updates the text on the screen automatically to reflect the new scroll position, as does the `TEScroll` function. The destination rectangle is offset by the amount scrolled. When the edit structure is longer than the text it contains, `TEPinScroll` displays up to the last line of text inclusive, and not beyond it.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEReplaceStyle

Replaces any character attributes in the current selection range that match the specified existing character attributes with the specified new character attributes. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEReplaceStyle (
   short mode,
   const TextStyle *oldStyle,
   const TextStyle *newStyle,
   Boolean fRedraw,
   TEHandle hTE
);
```

Parameters

mode

A selector that specifies which attributes to replace. It corresponds to any additive combination of the "Text Styling Constants" (page 50) for font, character style, type size, color, and so forth.

oldStyle

A pointer to a text style structure that specifies the current character attributes to search for in the selected text.

newStyle

A pointer to a text style structure that specifies the new attributes to be set. This structure contains the character attributes to be applied to the current selection range based on the value of `mode`.

fRedraw

A flag that specifies whether or not TextEdit should immediately redraw the text to reflect the attribute changes. A value of `FALSE` delays redrawing until another event forces the update. A value of `TRUE` causes the text to be redrawn immediately using the new character attributes.

hTE

A handle to the multistyled edit structure containing the text selection whose character attributes are to be changed.

Discussion

The `TEReplaceStyle` function replaces any attribute in the current selection range that matches the attribute specified by `oldStyle` with that given by `newStyle`. Only the character attributes specified by `mode` are affected.

Attribute changes are made directly to the style elements (`STElement`) within the style table itself (`TEStyleTable`). If you specify the value `doAll` for the `mode` parameter, `newStyle` replaces `oldStyle` outright. The `TEReplaceStyle` function has no effect on a monostyled edit structure.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEScrapHandle

Returns a handle to the TextEdit private scrap. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
Handle TEScrapHandle (
   void
);
```

Return Value

A handle to the TextEdit private scrap. See the Mac Types documentation for a description of the `Handle` data type.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEScroll

Scrolls the text within the view rectangle of the specified edit structure by the designated number of pixels. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEScroll (
   short dh,
   short dv,
   TEHandle hTE
);
```

Parameters

dh

The distance in pixels that the text is to be scrolled horizontally. A positive value moves the text to the right; a negative value moves the text to the left.

dv

The distance in pixels that the text is to be scrolled vertically. A positive value moves the text down; a negative value moves the text up.

hTE

A handle to the edit structure whose text is to be scrolled.

Discussion

The `TEScroll` function updates the text on the screen automatically to reflect the new scroll position. The destination rectangle is offset by the amount scrolled. The `TEScroll` and `TEPinScroll` functions behave the same, except that `TEPinScroll` stops scrolling when the last line of text is scrolled into view.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TESelView

Ensures, once automatic scrolling has been enabled by a call to the `TEAutoView` function or through the `TEFeatureFlag` function, that the selection range is visible, scrolling it into the view rectangle if necessary. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESelView (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure containing the text selection range.

Discussion

The top left part of the selection range is scrolled into view. If the text is displayed in a rectangle that is not high enough, automatic scrolling can cause text to appear to flicker. If automatic scrolling is disabled, `TESelView` has no effect. For more information, see `TEFeatureFlag` (page 79).

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetAlignment

Sets the alignment of the specified text in an edit structure so that it is centered, right aligned, or left aligned, or aligned according to the line direction. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetAlignment (
   short just,
   TEHandle hTE
);
```

Parameters

just

The alignment for the specified text.The default value of the `just` field of the edit structure is `teFlushDefault`. This means that text alignment is based on the primary line direction which is set by default according to the system script.

For a description of the values you can use in this parameter, see "Text Alignment Constants" (page 46).

hTE

A handle to the edit structure containing the text.

Discussion
For languages that are read from right to left, text is right aligned by default. For languages that are read from left to right, text is left aligned by default. If you change the alignment, call the Window Manager function `InvalRect` after `TESetAlignment` to redraw the text with the new alignment.

TextEdit does not support justified alignment. To draw justified text, use the QuickDraw Text functions.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetClickLoop

Installs the address of the application-supplied click loop function in the `clikLoop` field of the edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetClickLoop (
   TEClickLoopUPP clikProc,
   TEHandle hTE
);
```

Parameters

clikProc

A universal procedure pointer to the customized click loop function.

hTE

A handle to the edit structure whose `clikLoop` field is to be modified.

Discussion

The `TESetClickLoop` function lets you replace the default click loop function. The `TEClick` function repeatedly calls the function that the click loop field points to as long as the user holds down the mouse button within the text of the view rectangle. The default click loop function scrolls only the text. However, you can provide a customized click loop function that scrolls the text and the scroll bars in tandem.

If automatic scrolling is enabled, the default click loop function checks to see if the mouse has been dragged out of the view rectangle; if it has, the function scrolls the text using `TEPinScroll` (page 91). The amount by which `TEPinScroll` scrolls the text vertically is determined by the `lineHeight` field of the edit structure for monostyled text and the `LHTable` for multistyled text.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TESetDoTextHook

Sets your do-text-hook callback to be used by TextEdit. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetDoTextHook (
   TEDoTextUPP value
);
```

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TESetFindWordHook

Sets your set-find-word-hook callback to be used by TextEdit. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetFindWordHook (
   TEFindWordUPP value
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetRecalcHook

Sets your recalculation callback to be used by TextEdit. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetRecalcHook (
   TERecalcUPP value
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetScrapHandle

Sets a handle to the TextEdit private scrap. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetScrapHandle (
   Handle value
);
```

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetScrapLength

Sets the size of the TextEdit private scrap to the specified number of bytes. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetScrapLength (
   long length
);
```

Parameters

length

The size of the private scrap, in bytes.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TESetSelect

Sets the selection range (or denotes the insertion point) within the text of the specified edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetSelect (
   long selStart,
   long selEnd,
   TEHandle hTE
);
```

Parameters

selStart

The byte offset at the start of the text selection range. The `selStart` field can range from 0 to 32767.

If `selStart` equals `selEnd`, the new selection range is an insertion point, and a caret is displayed. If `selEnd` is anywhere beyond the last character of the text, `TESetSelect` uses the first position past the last character.

selEnd

The byte offset at the end of the text selection range. The `selEnd` field can range from 0 to 32767.

hTE

A handle to the edit structure.

Discussion
The `TESetSelect` function removes highlighting of the old selection range and highlights the new one.

When only the Roman script system is used, the selection range is always displayed and highlighted as a continuous range of text. However, when one or more script systems requiring mixed-directional display of text are installed, a continuous sequence of characters in memory may appear as a discontinuous selection when displayed.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
TextEdit.h

TESetStyle

Sets new character attributes, in the specified edit structure, for the current selection range. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetStyle (
   short mode,
   const TextStyle *newStyle,
   Boolean fRedraw,
   TEHandle hTE
);
```

Parameters

mode

A selector that specifies which character attributes are to be changed. The value for mode can be any additive combination of the mode constants for font, style, type size, color, and so forth. It corresponds to any additive combination of the "Text Styling Constants" (page 50) for font, character style, type size, color, and so forth.

The value of mode specifies which existing character attributes are to be changed to the new character attributes specified by newStyle. If doToggle is specified along with doFace and if an attribute specified in the given newStyle parameter exists across the entire selected range of text, then TESetStyle removes that attribute. Otherwise, if the attribute doesn't exist across the entire selection range, all of the selected text is set to include that character attribute.

newStyle

A pointer to a structure of type TextStyle that specifies the new attributes to be set. This structure contains the character attributes to be applied to the current selection range based on the value of mode.

fRedraw

A flag that specifies whether or not TextEdit should immediately redraw the affected text to reflect the new character attribute changes. A value of TRUE causes the text to be redrawn immediately. Line breaks, line heights, and line ascents are recalculated. A value of FALSE delays redrawing until another event forces the update.

If the fRedraw parameter is set to TRUE, TextEdit redraws the current selection range using the new character attributes, recalculating line breaks, line heights, and line ascents.

If the fRedraw parameter is set to FALSE, TextEdit does not redraw the text or recalculate line breaks, line heights, and line ascents. Consequently, when you call a function that uses any of this information, such as TEGetHeight (which returns a total height between two specified lines), it does not reflect the new character attributes set with TESetStyle. Instead, the function uses the information that was available before TESetStyle was called. To update this information, call the TECalText (page 72) function. To be certain that the new information is always reflected, call TESetStyle with the fRedraw parameter set to TRUE.

hTE

A handle to the multistyled edit structure containing the selected text.

Discussion
The TESetStyle function has no effect on a monostyled structure.

If you call the `TESetStyle` function when the value of the `selStart` field of an edit structure equals the value of the `selEnd` field (specifying an insertion point), TextEdit stores the input character attributes in the null scrap structure pointed to by the null style handle.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetStyleHandle

Sets an edit structure's style handle, which is stored in the `txFont` and `txFace` fields. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetStyleHandle (
   TEStyleHandle theHandle,
   TEHandle hTE
);
```

Parameters

`theHandle`
The style handle to be set in the combined `txFont` and `txFace` fields of the specified edit structure.

`hTE`
A handle to the edit structure.

Discussion
The `TESetStyleHandle` function has no effect on monostyled edit structures.

Your application should always use `TESetStyleHandle` rather than manipulate the fields of the edit structure directly.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TESetText

Incorporates a copy of the specified text into the designated edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TESetText (
   const void *text,
   long length,
   TEHandle hTE
);
```

Parameters

text

A pointer to the text to be copied and incorporated.

length

The number of characters in the text to be incorporated.

hTE

A handle to the edit structure into which the text is to be copied.

Discussion

The function `TESetText` lets you incorporate existing text into the text buffer of an edit structure. The function copies the specified text into the existing `hText` handle of the edit structure, resizing the buffer, if necessary it doesn't bring in the original text. The copied text is wrapped to the destination rectangle, and its `lineStarts` and `nLines` fields are calculated accordingly. The selection range is set to an insertion point at the end of the incorporated text. The `TESetText` function does not display the copied text on the screen. To do this, call `TEUpdate`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEStyleInsert

Inserts the specified text immediately before the selection range or the insertion point in the edit structure's text and applies the specified character attributes to the text, redrawing the text if necessary. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEStyleInsert (
   const void *text,
   long length,
   StScrpHandle hST,
   TEHandle hTE
);
```

Parameters

text

A pointer to the text to be inserted.

length

The length, in bytes, of the text to be inserted.

hST

A handle to the style scrap structure containing the character attribute information to be applied to the inserted text.

hTE

A handle to the edit structure into which the text is to be inserted.

Discussion
You should create your own style scrap structure, specifying the character attributes to be inserted and applied to the text, and pass its handle to `TEStyleInsert` as the value of the `hST` parameter. The character attributes are copied directly into the style structure's (`TEStyleRec`) style table.

The `TEStyleInsert` function does not affect the current selection range.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEStyleNew

Creates a multistyled edit structure and allocates a handle to it. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
TEHandle TEStyleNew (
   const Rect *destRect,
   const Rect *viewRect
);
```

Parameters
destRect

A pointer to the destination rectangle for the new edit structure, specified in the local coordinates of the current graphics port. This is the area in which text is laid out.

viewRect

A pointer to the view rectangle for the new edit structure, specified in the local coordinates of the current graphics port. This is the area of the window in which text is actually displayed.

Return Value
A handle to the newly created edit structure. Your application needs to store the handle to the edit structure that is returned; many functions require it as an input parameter. See the description of the `TEHandle` data type.

Discussion
A multistyled edit structure contains text whose attributes, including font, size, and style, can vary from character to character. Always use the `TEStyleNew` function to create an edit structure for text that uses varying character attributes. The `TEStyleNew` function sets the `txSize`, `lineHeight`, and `fontAscent` fields of the edit structure to –1, allocates a style structure, and stores a handle to the style structure in the `txFont` and `txFace` fields. The `TEStyleNew` function creates and initializes a null scrap that is used by TextEdit functions throughout the life of the edit structure.

Call `TEStyleNew` once for every edit structure you want allocated. Your application needs to store the handle to the edit structure that is returned; many functions require it as an input parameter.

If your application contains more than one window where text editing occurs, you need to create an edit structure for each window.

Before this function is called, the window must be in the current graphics port.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEStylePaste

Pastes text and its associated character attribute information from the desk scrap into the edit structure's text at the insertion point—if the current selection range is an insertion point—or it replaces the current selection range. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEStylePaste (
   TEHandle hTE
);
```

Parameters

hTE

A handle to the edit structure into which the text is to be pasted.

Discussion
When you call `TEStylePaste` and there is no character attribute information associated with text in the desk scrap, `TEStylePaste` first checks the null scrap. If the null scrap contains character attribute information, this is used. If the null scrap is empty, `TEStylePaste` gives the text the same attributes as those of the first character of the replaced selection range or that of the preceding character if the selection is an insertion point.

For a monostyled edit structure, `TEStylePaste` pastes the text only; there is no associated character attribute information because all the text uses the same attributes.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TETextBox

Draws the indicated text in a given rectangle, with the specified alignment. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TETextBox (
   const void *text,
   long length,
   const Rect *box,
   short just
);
```

Parameters

text

A pointer to the text to be drawn.

length

The number of bytes comprising the text.

box

A pointer to the rectangle where the text is to be drawn. The rectangle is specified in local coordinates (of the current graphics port) and must be at least as wide as the first character drawn. (A good rule of thumb is to make the rectangle at least 20 pixels wide.

just

The kind of justification (alignment) used for the specified text.

Discussion

The `TETextBox` function provides you with an easy way to display static text to a user. It creates its own monostyled edit structure, which it deletes when finished with it, so you cannot edit the text it draws. The `TETextBox` function breaks a line of text correctly. You can specify how text is aligned in the box using any of the "Text Alignment Constants" (page 46).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

TEToScrap

Copies the contents of the TextEdit private scrap to the desk scrap. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
OSErr TEToScrap (
   void
);
```

Return Value

A result code. See "TextEdit Result Codes" (page 51).

Discussion

You use the `TEToScrap` function to move monostyled text across applications or between an application and a desk accessory. Call the Scrap Manager function `ZeroScrap` to initialize the desk scrap or clear its contents before calling `TEToScrap`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEUpdate

Draws the specified text within a given update rectangle. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEUpdate (
   const Rect *rUpdate,
   TEHandle hTE
);
```

Parameters

rUpdate

The update rectangle, given in the coordinates of the current graphics port, where the specified text is to be drawn.

hTE

A handle to the edit structure containing the text to be drawn.

Discussion
Call `TEUpdate` every time the Event Manager function `WaitNextEvent` reports an update event for a text editing window—after you call the Window Manager function `BeginUpdate` , and before you call the `EndUpdate` function. You also need to erase the update region with the `EraseRect` function. If you don't, the caret can sometimes remain visible when the window is deactivated.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`TextEdit.h`

TEUseStyleScrap

Assigns new character attributes to the specified range of text in the designated edit structure. (Deprecated in Mac OS X v10.4. Use Multilingual Text Engine instead; see *Handling Unicode Text Editing With MLTE*.)

```
void TEUseStyleScrap (
   long rangeStart,
   long rangeEnd,
   StScrpHandle newStyles,
   Boolean fRedraw,
   TEHandle hTE
);
```

Parameters

rangeStart

The offset of the first character in the text of the edit structure to which the character attributes are to be applied.

rangeEnd

The offset of the last character in the text of the edit structure to which the character attributes are to be applied.

newStyles

A handle to a style scrap structure. The style scrap structure contains the attributes to be applied to the specified range of text. If the value of `newStyles` is `NULL`, no action is performed. Each element in the style scrap structure contains a field that is the offset of the beginning of the element's character attributes. This field (`scrpStartChar`) defines the boundaries for the scrap's style runs.

Depending on the requirements of your application, you can create a style scrap structure directly and pass its handle to `TEUseStyleScrap` as the value of `newStyles` or you can use a style scrap structure created by `TEGetStyleScrapHandle`.

fRedraw

A flag that specifies whether TextEdit should immediately redraw the selection range using the new character attributes. If the `fRedraw` parameter is set to `TRUE`, the attributes are applied immediately to the specified range of text, and line breaks, line heights, and line ascents are recalculated. If `fRedraw` is set to `FALSE`, the new character attributes are not reflected in the view rectangle until the next update event occurs.

hTE

A handle to the edit structure containing the range of text to which the character attributes are to be applied. If the handle points to a monostyled edit structure (created using `TENew`), no action is performed.

Discussion

The `TEUseStyleScrap` function writes the character attribute information into the style structure's style table and updates the style run table.

Regardless of whether the text is redrawn, the current selection range is not changed; if characters are highlighted before `TEUseStyleScrap` is called, they remain highlighted after it is called. However, if characters within the current selection range also fall within the specified range of text, they are rendered in the new character attributes when the text is redrawn.

The `TEUseStyleScrap` function applies the first element's attributes to the characters from `rangeStart` up to the `scrpStartChar` field of the next element. The function terminates without error if it prematurely reaches the end of the range or if there are not enough scrap style elements to cover the whole range. In the latter case, the function applies the last set of character attributes in the style scrap structure to the remainder of the range.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`TextEdit.h`

Document Revision History

This table describes the changes to *TextEdit Reference*.

Date	Notes
2006-07-13	Made minor formatting changes.
2006-07-24	Added information about deprecated functions.
2005-07-07	Fixed declaration for TSMDialogRecord.
2003-04-23	Minor corrections.
2003-04-10	Added abstracts for numerous functions, data types, and enumerations. Moved unsupported functions to an appendix.
2002-10-31	Updated formatting.

Document Revision History

Index

I

L

N

S

T

W