Text Parameters

Outline Highlighting

Use the teFOutlineHilite constant in the feature parameter of **TEFeatureFlag** to enable outline highlighting. If a highlighted region exists in an edit record in an inactive window, then the highlighted region is outlined (or framed) when the window is in the background, a behavior similar to MPW selections. If the caret is in the window and the window is no longer active, the caret is then drawn in a gray pattern so that it appears dimmed.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 record, redraws the caret or highlighted region, and then immediately restores the hooks to their previous addresses.

Text Buffering

Use the teFTextBuffering constant in the feature parameter of **TEFeatureFlag** to perform text buffering. Text buffering can be enabled for performance improvements, especially with double-byte scripts. **TextEdit** buffers each **TEKey** input of a graphic character. The entire buffer will then be inserted at one time if any **TextEdit** routine is called other than **TEKey** for another graphic character. This includes any routines that handle a mouse-down event, a style change, font and keyboard synchronization, the input of a nongraphic character, or a call to the **TEIdle** procedure. The buffer is dumped before this routine is handled.

> **Warning:** This buffer is a global buffer (and differs from **TEKey**'s internal double-byte buffer) and is used across all active edit records. These records may be in a single application or in multiple applications. Exercise care when you enable **TEFeatureFlag**'s text-buffering capability in more than one active record; otherwise, the bytes that are buffered from one edit record may appear in another edit record. You also need to be sure that buffering is not turned off in the middle of processing a double-byte character.
>
> To guarantee the integrity of your record, it is important that you wait for an idle event before you disable buffering or enable buffering in a second edit record.

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 buffering characters in the new script.

> **Note:** If the text-buffering feature teFTextBuffering is enabled, your application must ensure that **TEIdle** is called before any pause of more than a few ticks-for example, before **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 may end up in the edit record of another application.

Inline Input

If your application follows the guidelines for inline input available from Macintosh Developer Technical Support, then you should set the flag useTextEditServices in the 'SIZE' resource in your application. This allows

inline input to work with your application. Inline input is a keyboard input method (often used for double-byte script systems) in which conversion from a phonetic to an ideographic representation of a character takes place at the current line position where the text is intended to appear. This allows the user to type text directly in the line as opposed to a special conversion window. If inline input is installed and the useTextEditServices flag in the 'SIZE' resource is set, inline input sets **TextEdit**'s teFUseTextServices feature bit whenever an edit record is created. This bit is not used by **TextEdit**.

Inline input checks the teFUseTextServices bit during text editing to determine if an inline session should begin. If you want to disable inline input for a particular edit record, your application can clear this bit after the edit record is created. You can also clear this bit to disable inline input temporarily and then restore it, but the edit record should always be deactivated before the state of the bit is changed.

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.

> **Note:** You must deactivate an edit record before changing the state of the feature bits or any fields in the edit record.

When an inline edit session begins, inline input also sets the teFInlineInput bit to provide the following features so that inline input will work 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
- 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 record should always be deactivated before the state of the teFInlineInput bit is changed.

> **Warning:** If you clear the teFUseTextServices bit and you set the teFInlineInput bit, inline input is disabled, but your application retains the features listed above.