
Understanding Text Input and the Text Services Manager in Carbon

(Legacy)





Apple Inc.
© 2008 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Carbon, Mac, Mac OS, and Macintosh are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY

DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction	Introduction to Understanding Text Input and the Text Services Manager in Carbon 7
---------------------	---

Chapter 1	Understanding Text Input and the Text Services Manager 9
------------------	---

What's New for Applications With TSM and Text Input in Carbon	9
Text Input Event Handling for Applications With Carbon	11
Carbon Text Input Events and Apple Events	11
Text Input Event Routing	13
Handling Unicode and Keyboard Input in Carbon	13
What's New for Text Service Components With TSM and Text Input in Carbon	14
Event Handling for Text Service Components With Carbon	15

Document Revision History	17
----------------------------------	-----------

Tables

Chapter 1 **Understanding Text Input and the Text Services Manager 9**

Table 1-1 Text input events 11

Introduction to Understanding Text Input and the Text Services Manager in Carbon

Important: This text input management system described in this document is deprecated as of Mac OS X v10.6. Please see *Input Method Kit Framework Reference* and *NSTextInputClient Protocol Reference*, which describe the replacement technologies.

The **Text Services Manager** (“TSM”) provides an environment for applications to use non-application-specific text services. The Text Services Manager handles communication between client applications that request text services and the software modules, known as text service components, that provide them. The Text Services Manager exists so that these two types of programs can work together without needing to know anything about one another’s internal structures or identities. The Text Services Manager presents separate programming interfaces to the features it provides for client applications and for text service components.

A **client application** is a text-processing program that uses the Text Services Manager to request a service from a text service component. To accomplish this, a client application needs to make specific Text Services Manager calls during execution.

A **text service component** is a utility program that uses the Text Services Manager to provide a text service to an application. Text service components are registered components with the Component Manager. Text services can include many different types of specific text-handling tasks, including spell-checking, hyphenation, and handling the input of complex text.

The most prevalent category of text services are those that handle the entry of complex text, that is, **input methods**. A typical example of an input method is a service that converts keyboard input into text that cannot be directly entered via a keyboard. Text input in Japanese, Chinese, Korean, or Unicode usually requires an input method.

Note that application support for full Unicode input depends on the application’s adoption of the Text Services Manager. Applications that do not adopt the Text Services Manager’s input model only receive text input from partial Unicode sources, that is, only those input sources whose output can be converted to a given Mac encoding.

INTRODUCTION

Introduction to Understanding Text Input and the Text Services Manager in Carbon

Understanding Text Input and the Text Services Manager

This document presents the information needed to appropriately handle keyboard input in Carbon, to create Carbon text service components for Mac OS X, and to revise Carbon client applications of text services. It discusses keyboard input, Carbon text input events, Unicode input, and the changes to the Text Services Manager (“TSM”), in the context of Carbon and Mac OS X.

If you have a Carbon program that uses text services or if you are interested in handling raw keyboard events (that is, if you must do key filtering or key input validation), you should read the sections “[What’s New for Applications With TSM and Text Input in Carbon](#)” (page 9) and “[Text Input Event Handling for Applications With Carbon](#)” (page 11).

If you have a text service component, see “[What’s New for Text Service Components With TSM and Text Input in Carbon](#)” (page 14) and “[Event Handling for Text Service Components With Carbon](#)” (page 15).

This document assumes some prior knowledge of the Text Services Manager, text input, and keyboard events on the Mac OS. If you would like background information, please see the following sources:

- For an introduction to the Text Services Manager, see the chapter “Text Services Manager” in *Inside Macintosh: Text* at <http://developer.apple.com/documentation/mac/Text/Text-409.html>
- For a description of the Text Services Manager programming interface, see *Inside Carbon: Text Services Manager Reference* at http://developer.apple.com/documentation/Carbon/Reference/Text_Services_Manager/index.html.
- For a detailed description of Carbon text input events, see the CarbonLib Universal Interfaces file `CarbonEvents.h`.

What’s New for Applications With TSM and Text Input in Carbon

Adapting your application for Carbon is not mandatory, but it is recommended, both for increased compatibility with text services and for the enhanced performance of your program. As you read through this section, you may find that the changes discussed below simplify your use of the Text Services Manager and/or your handling of text input.

If your program directly handles keyboard input, you can read the list that follows for highlights of the changes in handling text input with Carbon.

If your application currently supports the Text Services Manager, the following list provides an overview and checklist for revising your application for the Text Services Manager in Carbon. If your application does not currently support the Text Services Manager, the list supplements the information given in the Text Services Manager chapter of *Inside Macintosh: Text*.

- Carbon provides a suite of text input events (corresponding to the text service Apple events) that supersede other mechanisms for handling text input. In most cases, if your application directly handles raw keyboard input, whether or not it uses the Text Services Manager, it should use Carbon text input events over raw keyboard events. For more information, see [“Handling Unicode and Keyboard Input in Carbon”](#) (page 13).
- At minimum, all application developers should consider providing a handler for the Carbon text input event `kEventTextInputOffsetToPos` (or the corresponding Apple event, `kOffset2Pos`). Handling `kEventTextInputOffsetToPos` (or `kOffset2Pos`) is recommended not only when there is not a currently active inline input session, but even if your application does not use the Text Services Manager and does not currently provide any other text input event handlers at all. By providing a handler that simply returns the position of the insertion point when the requested offset is 0, it makes your application more compatible with text services now and in the future.
- The `kEventTextInputOffsetToPos` Carbon text input event has an additional optional parameter (`kEventParamTextInputReplyFMFont = 'trfm'`) that the corresponding `kOffset2Pos` Apple event previously did not. (Note that the `kOffset2Pos` event now also has been upgraded to include a similar parameter, `keyAETSMTTextFMFont = 'ktxm'`.) This parameter consists of an `FMFont` value of type `UInt32`, allowing the event handler to return an `FMFont` value instead of an `FMFontFamily` value (or a `FOND` resource ID). The `FMFont` value is easier for ATSUI-based applications to obtain and more useful for ATSUI-based text services. For more information on using ATSUI to render Unicode text, see http://developer.apple.com/documentation/Carbon/CConceptual/ATSUI_Concepts/index.html.
- Carbon applications that support inline input from a text service (as opposed to automatically being provided a floating or “bottomline” input window by the Text Services Manager) should use the function `TSMSetInlineInputRegion`. The `TSMSetInlineInputRegion` function informs the Text Services Manager of the region occupied by an inline input session. If certain mouse events occur within this region, the Text Services Manager forwards these events to the current input method. If an application does not call this function, when an input method is active the Text Services Manager by default intercepts mouse events in the entire content region of the window that currently has user focus. The `TSMSetInlineInputRegion` function replaces the function `SetTSMCursor` for Carbon applications. For a description of the `TSMSetInlineInputRegion` function, see *Inside Carbon: Text Services Manager Reference*.
- Carbon applications do not need to pass events to the Text Services Manager as non-Carbon applications must. Therefore, Carbon applications need no longer use the functions `TSMEvent`, `TSMMenuSelect`, or `SetTSMCursor`. For more on how applications obtain events, see [“Text Input Event Handling for Applications With Carbon”](#) (page 11).
- Carbon clients of the Text Services Manager no longer need to call the functions `InitTSMAwareApplication` or `CloseTSMAwareApplication`. If a non-Carbon application uses any Text Services Manager routines, it must call these functions so the Text Services Manager can allocate necessary data for the application in its internal data structures, that is, so it can have a “TSM context.” Carbon automatically associates with each of its clients the data that is necessary for the Text Services Manager to interact with the application, so Carbon applications are automatically provided a TSM context.

Text Input Event Handling for Applications With Carbon

The Text Services Manager and text service components use text input events to give information to and request specific actions of client applications. Supporting full Unicode input and having inline input in your application depends on your installing handlers for these events. If your application does not handle text input events, the Text Services Manager automatically provides support only for partial Unicode input, via a floating or “bottomline” input window.

The following sections describe the various aspects of text input event handling for Carbon applications:

- “Carbon Text Input Events and Apple Events” (page 11)
- “Text Input Event Routing” (page 13)
- “Handling Unicode and Keyboard Input in Carbon” (page 13)

Carbon Text Input Events and Apple Events

Prior to Carbon, applications received text input events in the form of Apple events. With Carbon, the Text Services Manager now also provides Carbon text input events to applications. Supporting Carbon text input events is optional, but highly recommended. The performance advantages of using Carbon events over Apple events are significant. In addition, unlike Apple events, which are always targeted at the application level, Carbon events also can be targeted at the level of a window or even a control. If you have an existing Text Services Manager client application, revising it to use Carbon events is made simpler by the fact that it can be done piecemeal—you can change one, some, or all of its existing Apple event handlers at any time. [Table 2-1](#) (page 11) presents the Carbon text input events and notes the Apple events to which they correspond. (For a detailed description of the Carbon text input events, see the CarbonLib Universal Interfaces file `CarbonEvents.h`.)

Table 1-1 Text input events

Event description	Comments	Carbon event— <code>kEventClassTextInput</code>	Apple event— <code>kTextServiceClass</code>
Create or update text in an active input area	Must be supported for inline input	<code>kEventTextInput</code> <code>UpdateActiveInputArea</code>	<code>kUpdateActiveInputArea</code>
Receive Unicode text generated from a keyboard layout or via the TSM bottomline input window		<code>kEventTextInput</code> <code>UnicodeForKeyEvent</code>	<code>kUnicodeNotFromInputMethod</code>

Event description	Comments	Carbon event— <code>kEventClassTextInput</code>	Apple event— <code>kTextServiceClass</code>
Help position items on the screen by converting a byte offset in characters in the active input area to global coordinates	Supporting this event is recommended even for those applications that do not use inline input; to do so, provide a handler to this event that simply returns the position of the insertion point when the requested offset is 0	<code>kEventTextInputOffsetToPos</code>	<code>kOffset2Pos</code>
Help track cursor movements by converting global coordinates to a byte offset in characters in the active input area		<code>kEventTextInputPosToOffset</code>	<code>kPos2Offset</code>
Show or hide a floating input window	Most applications do not need to support this event; this event must be supported only by applications that provide their own bottomline input window	<code>kEventTextInputShowHideBottomWindow</code>	<code>kShowHideInput Window</code>
Return the current text selection or the text on either side of the current insertion point		<code>kEventTextInputGetSelectedText</code>	<code>kGetSelected Text</code>

The `kEventTextInputUnicodeForKeyEvent` event, by its very nature, always contains Unicode. For other Carbon text input events that contain text, a client application informs the Text Services Manager of what kind of encoding to use for these events when the application creates a new TSM document and specifies its desired interface type(s). For example, an application can request that its text be delivered in the Unicode encoding by creating a TSM document with the `kUnicodeDocument` signature.

Text Input Event Routing

The Text Services Manager mediates the routing of text input events. In the case of text service–generated events, the component calls the Text Services Manager function `SendTextInputEvent` to initiate the routing of the event. Then, the Text Services Manager delivers the event in the appropriate form for your application.

In general, the Text Services Manager tries to route events in their most optimal form. In Mac OS X, since Carbon events are more efficient than Apple events, the Text Services Manager first attempts to pass a text input event to an application as a Carbon event. The Text Services Manager routes Carbon text input events to the current user focus—typically a control, window, or application.

If the Carbon text input event is not handled, typically the Text Services Manager then generates the Apple event that corresponds to the unhandled Carbon event and sends that. And, if neither the Carbon text input event nor the corresponding Apple event is handled, then typically the event simply remains unhandled. An exception to these rules is the event `kEventTextInputUnicodeForKeyEvent`; see “[Handling Unicode and Keyboard Input in Carbon](#)” (page 13) for more details.

Handling Unicode and Keyboard Input in Carbon

To receive text input, applications must support the two keyboard-related text input events. The first, the `kEventTextInputUpdateActiveInputArea` Carbon event (or the Apple event `kUpdateActiveInputArea`), delivers text to your application that is generated by an input method in an inline input session. The second, the `kEventTextInputUnicodeForKeyEvent` Carbon event (or the Apple event `kUnicodeNotFromInputMethod`), delivers text to your application that is derived from a keyboard layout or via the Text Services Manager’s bottomline window. Especially if you want your client application to always receive Unicode characters, your application should handle the `kEventTextInputUnicodeForKeyEvent` text input event.

In order to avoid interference with input methods, providing a handler for the `kEventTextInputUnicodeForKeyEvent` event and examining its parameters is the preferred means on Carbon for directly examining keyboard input in general. Even if your application doesn’t support Unicode, it can still examine the `kEventTextInputUnicodeForKeyEvent` event’s parameters and extract Mac character codes, when they exist.

Only in special cases, such as if an application handles individual key presses or performs its own keyboard translation, might the application need to obtain a keyboard event prior to the event being processed by the Text Services Manager. In these cases, in Carbon you can obtain such events by installing Carbon event handlers for raw keyboard events. The Carbon Event Manager then dispatches the event to the application before it is sent to the Text Services Manager.

The Text Services Manager sends the `kEventTextInputUnicodeForKeyEvent` event to deliver Unicode text when the input source is either a keyboard layout or a text service via the Text Services Manager’s bottomline input window. Unlike other Carbon text input events, in the case of `kEventTextInputUnicodeForKeyEvent` the Text Services Manager falls back to sending the corresponding Apple event only if the application specifically requests Unicode (by creating a TSM document that has the `kUnicodeDocument` signature). So, if the application has requested Unicode, but it does not handle the `kEventTextInputUnicodeForKeyEvent` event, only then does the Text Services Manager convert the event to a `kUnicodeNotFromInputMethod` Apple event. Finally, if the application does not handle the `kUnicodeNotFromInputMethod` Apple event, then the Text Services Manager attempts to produce a stream of “classic” events, which the application can handle in its `WaitNextEvent` loop.

For more information on Unicode input, as of Mac OS 8.5, see *Inside Macintosh: Supporting Unicode Input* at http://developer.apple.com/documentation/Carbon/Conceptual/Supporting_Unicode_Input/index.html.

What's New for Text Service Components With TSM and Text Input in Carbon

With Mac OS X, text service components must be Carbon clients. (This is in contrast to Mac OS 8 and 9, where text service components cannot be Carbon clients, due to the potentially destabilizing effects of a component loading Carbon in the context of a non-Carbon application.) To help you in creating a Mac OS X-ready component, the following list provides a brief overview of the changes required:

- Unlike applications, text service components must be Carbon event-based for Mac OS X. Components are required to use Carbon events with Mac OS X in order to receive events targeted at their own user interface elements, such as the input method menu and its floating windows. Because on Mac OS X the Text Services Manager no longer calls the functions `TextServiceMenuSelect` or `SetTextServiceCursor`, the component must install Carbon event handlers on its user interface elements for these purposes. For more details, see “Event Handling for Text Service Components With Carbon” (page 15).
- Components should also use Carbon text input events when sending events. The function `SendTextInputEvent` is provided for this purpose. The sending of Apple events and the use of the pre-Carbon function `SendAEFromTSMComponent` is discouraged on Mac OS X.
- Text service components in Mac OS X must use Carbon Window Manager functions to create floating utility windows, rather than the Text Services Manager “service window” calls. For example, to create a floating input window, a Carbon text service component should use the Window Manager function `CreateNewWindow` (of class `kUtilityWindowClass`), rather than the Text Services Manager function `NewServiceWindow`, which is not available in Carbon. Note that, in this example, input methods should clear the `kWindowHideOnFullScreenAttribute` window attribute that is part of the default Carbon Window Manager behavior for utility windows, since it is likely that the input method would continue to be needed by the user, even if the application were to be put into full-screen mode.
- Unlike with Mac OS 8 and 9, instances of a single component that are running in different applications in Mac OS X do not share the same address space. Because of this, in Mac OS X, you may want to factor out some shared elements of your text service component into a separate process. For example, factoring out your dictionaries and global user interface elements ensures a continuity of state in your component's menus and windows among applications, as well as helping to reduce your memory requirements.
- Text service components for Mac OS X are assumed to use Unicode. The main requirement imposed by the Text Services Manager is that input methods communicate externally using Unicode text. However, it is assumed to be desirable or necessary for performance and other reasons that an input method performs its internal processing in Unicode and image Unicode text in its user interface. For more information, as of Mac OS 8.5, see *Inside Macintosh: Supporting Unicode Input* at http://developer.apple.com/documentation/Carbon/Conceptual/Supporting_Unicode_Input/index.html.

Event Handling for Text Service Components With Carbon

Unlike client applications, text service components both send and receive events. Text service components, via the Text Services Manager, send text input events to give information to and request specific actions of client applications. The component calls the Text Services Manager function `SendTextInputEvent` to initiate the routing of the event, which is delivered by the Text Services Manager in the form appropriate for the application. (Note that the sending of Apple events and the use of the pre-Carbon function `SendAEFromTSMComponent` is discouraged on Mac OS X.) For more information on text input events, see [“Text Input Event Handling for Applications With Carbon”](#) (page 11).

A text service component also receives certain events targeted at the current user focus from the Text Services Manager, which it intercepts on the component’s behalf. These events are of two types: keyboard events and mouse events that intersect an inline input session’s region. The Text Service Manager routes these events to the active input method and to any other text services associated with the active TSM document through calls to the function `TextServiceEventRef`. For a more detailed discussion of the `TextServiceEventRef` function and the `SendTextInputEvent` function, see *Inside Carbon: Text Services Manager Reference*.

The Text Services Manager works on behalf of components to intercept keyboard events (these are Carbon events of class `kEventClassKeyboard`) in two passes. First, the Text Services Manager intercepts the raw keyboard event and uses a keyboard-layout resource to translate the keypress into character data. The Text Services Manager stores the keyboard translation information in the keyboard event and dispatches it to the current user focus—such as a window, menu, or control. Then, if the event is not handled, the Text Services Manager sends the event to the text service component.

In addition to keyboard events, the Text Services Manager intercepts mouse events for routing to components. The Text Services Manager directly passes any mouse-click events (`kEventMouseDown`, `kEventMouseUp`, `kEventMouseDragged`) that intersect an inline input session’s region to the active component(s).

The Text Services Manager also intercepts mouse-moved events, however it does not pass them along in a direct form to components, as it does with mouse-click events. The Text Services Manager promotes `kEventMouseMoved` events, after the applicable control or window receives them, to the window-specific `kEventWindowCursorChange` event. Next, the Text Services Manager delivers the `kEventWindowCursorChange` event to the active input method, then, if it is not handled, to any other active text services. If the event is still not handled, the Text Services Manager finally passes the event to the current user focus. This event-dispatching process gives applications that need to see the low-level mouse-moved events a chance to see these events first, while providing a mechanism for text services and applications to act on these events without conflict. After completing this process, the Carbon Event Manager converts any `kEventWindowCursorChange` event that remains unhandled to a “classic” mouse-moved event for `WaitNextEvent` clients.

Document Revision History

This table describes the changes to *Understanding Text Input and the Text Services Manager in Carbon*.

Date	Notes
2008-09-30	Moved document to Legacy.
2002-12-11	Fixed formatting errors.
2001-05-01	Revision.

REVISION HISTORY

Document Revision History