Input Method Kit Framework Reference

Cocoa > Internationalization

2007-06-06

Apple Inc.
© 2004, 2007 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, Cocoa, Mac, Mac OS, and Objective-C 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

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Tables

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

The Input Method Kit Reference Collection

Framework	System/Library/Frameworks/InputMethodKit.framework
Header file directories	System/Library/Frameworks/InputMethodKit.framework/Headers
Declared in	IMKCandidates.h IMKInputController.h IMKInputSession.h IMKServer.h

The Input Method Kit, introduced in Mac OS X v10.5, provides a streamlined programming interface that lets you develop input methods with far less code than older Mac programming interfaces. It is fully integrated with the Text Services Manager. The Input Method Kit allows 32-bit applications to work with 64-bit applications.

The Input Method Kit provides classes and protocols for managing communication with client applications, candidates windows, and input method modes. Input methods supply text from a conversion engine (written in any language, such as C, C++, Objective-C, Python, and so on), key bindings and optional event handling, and information about your input method in an extended `Info.plist` file. You also have the option to provide menu items that support input-method-specific commands or preferences settings.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

The Input Method Kit Reference Collection

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Classes

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKCandidates Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	System/Library/Frameworks/InputMethodKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	InputMethodKit/IMKCandidates.h
Related sample code	NumberInput_IMKit_Sample

Overview

The `IMKCandidates` class presents candidates to users and notifies the appropriate `IMKInputController` object when the user selects a candidate. **Candidates** are alternate characters for a given input sequence. The `IMKCandidates` class supports using a candidates window in your input method; using `IMKCandidates` is optional. Not all input methods require them.

When you create an `IMKCandidates` object, you attach it to the `IMKServer` object for your input method. You then need to override the `IMKInputController` methods `candidateSelectionChanged:` and `candidateSelected:` as well as implement a candidates method in your delegate object. The `IMKInputController` subclass supplies candidates to the `IMKCandidates` object by implementing the candidates method. When you are ready to display a candidates window, call the candidates method to update candidates and to show the candidates window.

Tasks

Initializing a Candidates Window

`- initWithServer:panelType:` (page 14)
Returns the initialized `IMKCandidates` object.

Managing Selection Keys

`- setSelectionKeys:` (page 17)
Sets the selection keys for the candidates.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

- `selectionKeys` (page 15)
  Returns an array of `NSNumber` objects where each `NSNumber` object represents a virtual key code.
- `setSelectionKeysKeylayout:` (page 18)
  Sets the key layout that is used to map virtual key codes to characters.
- `selectionKeysKeylayout` (page 15)
  Returns the key layout that maps virtual key codes to selection keys.

Managing Window Visibility and Behavior

- `show:` (page 18)
  Shows the candidates window.
- `hide` (page 13)
  Hides a candidates window, if it is visible.
- `isVisible` (page 14)
  Returns whether or not the candidates window is visible.
- `setDismissesAutomatically:` (page 16)
  Sets the state of the flag that determines whether the candidates window dismisses automatically.
- `dismissesAutomatically` (page 13)
  Returns the state of the flag that determines whether the candidates window dismisses automatically.
- `updateCandidates` (page 19)
  Updates the candidates that are displayed in the candidates window.

Managing Window Type and Text Attributes

- `panelType` (page 14)
  Returns the style of the candidates window.
- `setPanelType:` (page 17)
  Sets the style of the candidates window.
- `setAttributes:` (page 16)
  Sets the style attributes for the candidates window.
- `attributes` (page 13)
  Returns a dictionary of the style attributes used for the candidates window..

Showing an Annotation Window

- `showAnnotation:` (page 19)
  Displays an annotation string in an annotation window.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Instance Methods

attributes

Returns a dictionary of the style attributes used for the candidates window..

```
- (NSDictionary *)attributes
```

Return Value
The dictionary that contains the keys and values for the styles.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setAttributes:` (page 16)

Declared In
`IMKCandidates.h`

dismissesAutomatically

Returns the state of the flag that determines whether the candidates window dismisses automatically.

```
- (BOOL)dismissesAutomatically
```

Return Value
`YES` if the candidates window dismisses automatically; otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setDismissesAutomatically:` (page 16)

Declared In
`IMKCandidates.h`

hide

Hides a candidates window, if it is visible.

```
- (void)hide
```

Availability
Available in Mac OS X v10.5 and later.

See Also
- `show:` (page 18)
- `isVisible` (page 14)

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
IMKCandidates.h

initWithServer:panelType:

Returns the initialized IMKCandidates object.

```
- (id)initWithServer:(IMKServer *)server panelType:(IMKCandidatePanelType)panelType
```

Parameters

server
The IMKServer object that manages the candidate and the panel type.

panelType
A panel type for the candidate window.

Return Value
The initialized IMKCandidates object.

Discussion
When an input method allocates an IMKCandidates object it should initialize that object by calling this method.

Availability
Available in Mac OS X v10.5 and later.

Declared In
IMKCandidates.h

isVisible

Returns whether or not the candidates window is visible.

```
- (BOOL)isVisible
```

Return Value
YES if the candidates window is visible; otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- hide (page 13)
- show: (page 18)

Declared In
IMKCandidates.h

panelType

Returns the style of the candidates window.

```
- (IMKCandidatePanelType)panelType
```


2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Return Value
A "IMKCandidatePanelType" (page 20) constant that represents the style of the candidates window.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setPanelType:` (page 17)

Declared In
`IMKCandidates.h`

selectionKeys

Returns an array of `NSNumber` objects where each `NSNumber` object represents a virtual key code.

```
- (NSArray *)selectionKeys
```

Return Value
The array of `NSNumber` objects. This array is an autorelease object. Do not release it unless you first retain it.

Discussion
Selection keys are keys that can be used to select one of the candidates. They are displayed next to the candidate that will be selected when the user types that key.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setSelectionKeys:` (page 17)

Declared In
`IMKCandidates.h`

selectionKeysKeylayout

Returns the key layout that maps virtual key codes to selection keys.

```
- (TISInputSourceRef)selectionKeysKeylayout
```

Return Value
The key layout in use. By default this is the key layout whose source id is `com.apple.keylayout.US`. This object is an autorelease object. Do not release it unless you first retain it.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setSelectionKeysKeylayout:` (page 18)

Declared In
`IMKCandidates.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

setAttributes:

Sets the style attributes for the candidates window.

```
- (void)setAttributes:(NSDictionary *)attributes
```

Parameters

attributes

A dictionary that contains keys and values for the styles to use. You can supply the keys and values listed in the following table:

Key	Value
NSFontAttributeName	An NSFont object. Setting the font attribute sets the font that is used to draw Candidates. It does not effect the selection keys which are always drawn in the same font. Note that to set the font size you should use this key/value pair.
IMKCandidatesOpacityAttributeName (page 21)	An NSNumber object that represents a floating-point value between 0.0 (transparent) and 1.0 (completely opaque. The default opacity is 1.0.
NSForegroundColorAttributeName	An NSColor object to use for the candidate text color. The default color is black.
NSBackgroundColorDocumentAttribute	An NSColor object to use for the background color behind the candidate text.

Availability
Available in Mac OS X v10.5 and later.

See Also
- attributes (page 13)

Declared In
IMKCandidates.h

setDismissesAutomatically:

Sets the state of the flag that determines whether the candidates window dismisses automatically.

```
- (void)setDismissesAutomatically:(BOOL)flag
```

Parameters

flag

YES to have the candidates window dismiss automatically; otherwise NO.

Discussion
By default, if the user presses the Return or Enter keys, the candidates are dismissed and a candidateSelected: message is sent to the input controller. You can call the setDismissesAutomatically: method, passing NO as the flag parameter to change the default dismissal behavior. The input controller still receives a candidatesSelected: message.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

When you set the flag to `NO`, an input method processes text input while dynamically updating the content of the candidates as the user inputs text. When a session deactivates, candidate window is hidden regardless of the state of the flag.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- dismissesAutomatically` (page 13)

Declared In
`IMKCandidates.h`

setPanelType:

Sets the style of the candidates window.

```
- (void)setPanelType:(IMKCandidatePanelType)panelType
```

Parameters

panelType
A "IMKCandidatePanelType" (page 20) constant that represents the style of the candidates window.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- panelType` (page 14)

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`IMKCandidates.h`

setSelectionKeys:

Sets the selection keys for the candidates.

```
- (void)setSelectionKeys:(NSArray *)keyCodes
```

Parameters

keyCodes
An array of `NSNumber` objects where each `NSNumber` object represents a virtual key code. The input controller maps these key codes to characters that are displayed either across the top of the candidates, if the candidates are laid out horizontally, or along the left edge of the candidates, if they are aligned vertically.

Discussion
Selection keys are keys that can be used to select one of the candidates. They are displayed next to the candidate that will be selected when the user types that key.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

The number of selection keys determines how many candidates are displayed per page. For example, if you pass an array of four key codes, four candidates are displayed per page. If you pass eleven key codes, eleven candidates are displayed. By default, the key codes are mapped using the keyboard layout whose source id is `com.apple.keylayout.US`. You can replace the default layout by calling `setSelectionKeysKeylayout:` (page 18). The default selection keys are the digits 1 through 9 or, in terms of key codes, 18, 19, 20, 21, 23, 22, 26, 28, and 25.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- selectionKeys` (page 15)

Declared In
`IMKCandidates.h`

setSelectionKeysKeylayout:

Sets the key layout that is used to map virtual key codes to characters.

```
- (void)setSelectionKeysKeylayout:(TISInputSourceRef)layout
```

Parameters

layout

The key layout to use.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- selectionKeysKeylayout` (page 15)

Declared In
`IMKCandidates.h`

show:

Shows the candidates window.

```
- (void)show:(IMKCandidatesLocationHint)locationHint
```

Parameters

locationHint

A "IMKCandidatesLocationHint" (page 20) constant that specifies the desired position of the candidates window. The Input Method Kit uses the hint to place the candidates window in a location that is in the vicinity of the hint location and ensures that the candidates window is fully visible.

Discussion
Your input method calls this method when it is appropriate during text conversion to display a list of candidates.

Availability
Available in Mac OS X v10.5 and later.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

See Also
- `hide` (page 13)
- `isVisible` (page 14)

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`IMKCandidates.h`

showAnnotation:

Displays an annotation string in an annotation window.

```
- (void)showAnnotation:(NSAttributedString *)annotationString
```

Parameters

annotationString
The string to display.

Discussion
An annotation string explains or comments on the candidate string in the candidates window. An annotation window is a small, borderless window that is aligned with the current candidates window. An input method calls `showAnnotation:` when the `candidateSelectionChanged:` method of the `IMKInputController` class is called, and the candidate string has annotations.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKCandidates.h`

updateCandidates

Updates the candidates that are displayed in the candidates window.

```
- (void)updateCandidates
```

Discussion
When you call this method, the Input Method Kit calls the candidates method of the `IMKInputController` class. Note that the candidates list is updated, but the visible state of the window does not change. In other words, if the window is hidden, it remains hidden. If the window is visible, it remains visible.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`IMKCandidates.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Constants

IMKCandidatePanelType

Types of candidates windows provide by the Input Method Kit.

```
enum {   kIMKSingleColumnScrollingCandidatePanel = 1,
        kIMKScrollingGridCandidatePanel = 2,
         kIMKSingleRowSteppingCandidatePanel = 3 };
typedef NSUInteger IMKCandidatePanelType;
```

Constants

`kIMKSingleColumnScrollingCandidatePanel`

A window that displays one column and can scroll if necessary.

Available in Mac OS X v10.5 and later.

Declared in `IMKCandidates.h`.

`kIMKScrollingGridCandidatePanel`

A window that displays a grid and can scroll if necessary.

Available in Mac OS X v10.5 and later.

Declared in `IMKCandidates.h`.

`kIMKSingleRowSteppingCandidatePanel`

A window that displays a single row.

Available in Mac OS X v10.5 and later.

Declared in `IMKCandidates.h`.

Declared In

`IMKCandidates.h`

IMKCandidatesLocationHint

Hints that suggest where to place the candidates window.

```
enum {   kIMKLocateCandidatesAboveHint = 1,
        kIMKLocateCandidatesBelowHint = 2,
         kIMKLocateCandidatesLeftHint = 3,
         kIMKLocateCandidatesRightHint = 4
};typedef NSUInteger IMKCandidatesLocationHint;
```

Constants

`kIMKLocateCandidatesAboveHint`

Place the candidates window above the start of the current text selection.

Available in Mac OS X v10.5 and later.

Declared in `IMKCandidates.h`.

`kIMKLocateCandidatesBelowHint`

Place the candidates window below the start of the current text selection.

Available in Mac OS X v10.5 and later.

Declared in `IMKCandidates.h`.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

`kIMKLocateCandidatesLeftHint`
: Place the candidates window to the left of the current text selection.

  Available in Mac OS X v10.5 and later.

  Declared in `IMKCandidates.h`.

`kIMKLocateCandidatesRightHint`
: Place the candidates window to the right of the current text selection.

  Available in Mac OS X v10.5 and later.

  Declared in `IMKCandidates.h`.

Discussion
The Input Method Kit uses the hint to place the candidates window in a location that is in the vicinity of the hint location, but that also ensures that the candidates window is fully visible.

Declared In
`IMKCandidates.h`

IMKCandidatesOpacityAttributeName

The opacity level for a candidates window.

```
extern const NSString* IMKCandidatesOpacityAttributeName;
```

Constants
`IMKCandidatesOpacityAttributeName`
: The opacity attribute for a candidates window. The associated value must be an `NSNumber` object that represents a value from 0 to 1.

  Available in Mac OS X v10.5 and later.

  Declared in `IMKCandidates.h`.

Declared In
`IMKCandidates.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKInputController Class Reference

Inherits from	NSObject
Conforms to	IMKMouseHandling IMKStateSetting NSObject (NSObject)
Framework	System/Library/Frameworks/InputMethodKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	InputMethodKit/IMKInputController.h
Related sample code	NumberInput_IMKit_Sample

Overview

The `IMKInputController` class provides a base class for custom input controller classes. The `IMKServer` class, which is allocated in the main function of an input method, creates an input controller object for each input session created by a client application. For every input session there is a corresponding `IMKInputController` object.

An `IMKInputController` object controls text input on the input method side. It manages events and text from the applications and converted text from the input method engine. `IMKInputController` implements fully the `IMKStateSetting` and `IMKMouseHandling` protocols. Typically you do not need to override this class, but you do need to provide a delegate object that implements the methods that your are interested in. The `IMKInputController` versions of the protocol methods check whether the delegate object implements a method, and calls the delegate version if it exists.

Tasks

Initializing an Input Controller

`- initWithServer:delegate:client:` (page 28)
Initializes the input control by setting the delegate.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Working with Ranges

- `compositionAttributesAtRange:` (page 27)
  Returns a dictionary of text attributes.
- `selectionRange` (page 30)
  Returns where the range of the selection that should be placed inside marked text.
- `replacementRange` (page 30)
  Returns the range in the client document that the text should replace.
- `markForStyle:atRange:` (page 29)
  Returns a dictionary of text attributes that can mark a range of an attributed string to send to a client.

Managing the Delegate

- `delegate;` (page 27)
  Returns the delegate for input controller object.
- `setDelegate:` (page 31)
  Sets the delegate for input controller object.

Getting the Client and Server Objects

- `server` (page 31)
  Returns the server object that manages the input controller.
- `client` (page 26)
  Returns the client object associated with the input controller.

Tracking Selections

- `annotationSelected:forCandidate:` (page 25)
  Sends the selected candidate string and annotation string to the input controller.
- `candidateSelectionChanged:` (page 26)
  Informs an input controller that the current candidate selection in the candidate window has changed.
- `candidateSelected:` (page 26)
  Informs an input controller that a new candidate is selected.

Managing Composition

- `updateComposition` (page 31)
  Informs the input controller that the composition has changed.
- `cancelComposition` (page 25)
  Stops the current composition and replaces marked text with the original text.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Hiding the User Interface

- `hidePalettes` (page 28)
  Informs an input method that it should close any visible user interface.

Working with Custom Commands

- `doCommandBySelector:commandDictionary:` (page 27)
  Passes commands that are not generated as part of the text input process.
- `menu` (page 30)
  Returns a menu of commands that are specific to an input method.

Instance Methods

annotationSelected:forCandidate:

Sends the selected candidate string and annotation string to the input controller.

```
- (void)annotationSelected:(NSAttributedString*)annotationString
    forCandidate:(NSAttributedString*)candidateString
```

Parameters

annotationString
  The annotation string associated with the candidate.

candidateString
  The candidate string that the user moved to.

Discussion
This method is called when the user moves to a candidate.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

cancelComposition

Stops the current composition and replaces marked text with the original text.

```
- (void)cancelComposition
```

Discussion
This method calls the method `originalString:` to obtain the original text and sends that text to the client using a call to the `IMKTextInput` protocol method `insertText:replacementRange:` (page 58)

Availability
Available in Mac OS X v10.5 and later.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
`IMKInputController.h`

candidateSelected:

Informs an input controller that a new candidate is selected.

```
- (void)candidateSelected:(NSAttributedString*)candidateString
```

Parameters

candidateString
The changed candidate string.

Discussion
The candidate object is the user's final choice from the candidate window. The candidate window is closed before this method is called.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- candidateSelectionChanged:` (page 26)

Declared In
`IMKInputController.h`

candidateSelectionChanged:

Informs an input controller that the current candidate selection in the candidate window has changed.

```
- (void)candidateSelectionChanged:(NSAttributedString*)candidateString
```

Parameters

candidateString
The changed candidate string.

Discussion
Note this method is called to indicate user activity in the candidate window. The candidate object might not be the user's final selection.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- candidateSelected:` (page 26)

Declared In
`IMKInputController.h`

client

Returns the client object associated with the input controller.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
- (<IMKTextInput, NSObject>))client
```

Return Value
The client object. The returned object is an autoreleased object.

Discussion
The client object conforms to the `IMKTextInput` protocol.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

compositionAttributesAtRange:

Returns a dictionary of text attributes.

```
- (NSMutableDictionary*) compositionAttributesAtRange:(NSRange)range
```

Parameters

range
The range of text whose attributes you want to obtain.

Return Value
The dictionary of text attributes. The default implementation returns an empty dictionary.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

delegate;

Returns the delegate for input controller object.

```
- (id)delegate
```

Return Value
The delegate object. The returned object is an autoreleased object.

See Also
`- setDelegate:` (page 31)

doCommandBySelector:commandDictionary:

Passes commands that are not generated as part of the text input process.

```
- (void)doCommandBySelector:(SEL)aSelector
    commandDictionary:(NSDictionary*)infoDictionary
```


2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Parameters

aSelector

A selector that represents a command from the text input menu.

infoDictionary

A dictionary that contains two key-value pairs:

- `kIMKCommandMenuItemName` (page 48), whose value is an `NSMenuItem` object. That is, the item selected by the user.
- `kIMKCommandClientName` (page 48), whose value is the current client—`id<IMKTextInput, NSObject>`.

Discussion

The default implementation checks if the input controller object (that is, self) responds to the selector. If so, it sends the message `performSelector:withObject:` to the input controller class. The object parameter in that case is the `infoDictionary` parameter.

This method is called when a user selects a command from the text input menu. To support this, an input method must provide actions for each menu item that is placed in the menu. For example, `(void)menuAction:(id)sender`. Note that the sender in this instance is the info dictionary.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- menu` (page 30)

Declared In

`IMKInputController.h`

hidePalettes

Informs an input method that it should close any visible user interface.

```
- (void)hidePalettes
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

initWithServer:delegate:client:

Initializes the input control by setting the delegate.

```
- (id)initWithServer:(IMKServer*)server delegate:(id)delegate client:(id)inputClient
```

Parameters

server

The server object for the controller.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

delegate
The delegate object.

inputClient
The client object that will send messages to the controller using the server object. The client object must confirm to the `IMKTextInput` protocol.

Return Value
The initialized input controller object.

Discussion
Methods in the `IMKStateSetting` and `IMKMouseHandling` protocols that are implemented by the delegate object always include a client parameter. Methods in the `IMKInputController` class do not need to take a client because the `initWithServer:delegate:client:` method stores the client object you supply as an ivar when it initializes the `IMKInputController` object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

markForStyle:atRange:

Returns a dictionary of text attributes that can mark a range of an attributed string to send to a client.

```
- (NSDictionary*)markForStyle:(NSInteger)style atRange:(NSRange)range
```

Parameters

style
A style, which should be one of the following values: `kTSMHiliteSelectedRawText`, `kTSMHiliteConvertedText`, or `kTSMHiliteSelectedConvertedText`. See the `AERegistry.h` header file for the definition of these values.

range
The range (that is, a clause) to mark.

Return Value
The dictionary of text attributes. The returned object should be an autoreleased object.

Discussion
This utility function can be called by input methods to mark each range (i.e. clause) of marked text. T

The default implementation first calls the method `compositionAttributesAtRange:` (page 27) to obtain the additional attributes that an input method wants to include, such as font or glyph information. Then, it adds the appropriate underline and underline color information to the attributes dictionary for the style parameter. Finally it adds the style value as the dictionary value. The key for the style value is `NSMarkedClauseSegmentAttributeName`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

menu

Returns a menu of commands that are specific to an input method.

```
- (NSMenu*)menu
```

Return Value
The menu object. This object is an autoreleased object.

Discussion
This method is called whenever the menu needs to be drawn so that an input method can update the menu to reflect the current state.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- doCommandBySelector:commandDictionary:` (page 27)

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`IMKInputController.h`

replacementRange

Returns the range in the client document that the text should replace.

```
- (NSRange)replacementRange
```

Return Value
The range to replace.

Discussion
This method is called by `updateComposition` (page 31) to obtain the range in the client document where marked text should be placed. The default implementation returns an `NSRange` object whose location and length are `NSNotFound`. That indicates that the marked text should be placed at the current insertion point. Input methods that insert marked text somewhere other than at the current insertion point should override this method.

An example of an input method that might override this method would be one replaces words with synonyms. That input method would watch for certain words and when it detects such a word it would replaced the word by marked text that was a synonym of the word.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

selectionRange

Returns where the range of the selection that should be placed inside marked text.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
- (NSRange)selectionRange
```

Return Value
The range of the selection. This object should be an autoreleased object.

Discussion
This method is called by `updateComposition` (page 31) to obtain the selection range for marked text. The default implementation sets the selection range at the end of the marked text. You should override this method if your input method provides font or glyph information.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

server

Returns the server object that manages the input controller.

```
- (IMKServer*)server
```

Return Value
The server object. The returned object is an autoreleased object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

setDelegate:

Sets the delegate for input controller object.

```
- (void)setDelegate:(id)newDelegate
```

Parameters
`newDelegate`
The delegate object to set.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- delegate;` (page 27)

Declared In
`IMKInputController.h`

updateComposition

Informs the input controller that the composition has changed.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
- (void)updateComposition
```

Discussion
This method calls the protocol method `composedString:` to obtain the current composition. The current composition is sent to the client by a call to the method `setMarkedText:selectionRange:replacementRange:` (page 61).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKServer Class Reference

Inherits from	NSObject
Conforms to	IMKServerProxy NSObject (NSObject)
Framework	System/Library/Frameworks/InputMethodKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	InputMethodKit/IMKServer.h
Related sample code	NumberInput_IMKit_Sample

Overview

The `IMKServer` class manages client connections to your input method. When you write the main function for your input method, you create an `IMKServer` object. You should never need to override this class.

Tasks

Initializing a Server Object

- `initWithName:bundleIdentifier:` (page 34)
  Creates and returns a server object from property list information contained in the provided bundle.
- `initWithName:controllerClass:delegateClass:` (page 35)
  Creates and returns a server object initialized with the provided parameters.

Getting a Bundle for the Input Method

- `bundle` (page 34)
  Returns an `NSBundle` object for the input method.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Instance Methods

bundle

Returns an `NSBundle` object for the input method.

```
- (NSBundle*)bundle
```

Return Value
An `NSBundle` object that is either created from the bundle identifier contained in the server object, or from the main bundle. The returned object is an autoreleased object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKServer.h`

initWithName:bundleIdentifier:

Creates and returns a server object from property list information contained in the provided bundle.

```
- (id)initWithName:(NSString*)name bundleIdentifier:(NSString*)bundleIdentifier
```

Parameters

name
The name to initialize the server object with.

bundleIdentifier
The bundle identifier.

Return Value
An initialized server object.

Discussion
This method examines the `Info.plist` file for the entries shown in Table 3-1. The class names are loaded, but no classes are instantiated. Additionally, an `NSConnection` object is allocated and registered using the input method connection name supplied in the `Info.plist` file.

Table 3-1 Required entries in the `Info.plist` file

Key	Value
`LSBackgroundOnly`	The associated value is `1`, because input methods are background-only applications.
`InputMethodConnectionName`	A string that specifies an input method connection name that names the connection through which your input method services are published. The Input Method Kit uses this name to create an `NSConnection` object through which clients deliver text input.
`InputMethodServer-ControllerClass`	An input controller class.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Key	Value
`tsInputMethodIconFileKey`	An icon file name. The icon is used to display your input method in the International pane of System Preferences.
`tsInputMethod-CharacterRepertoireKey`	An array of one or more ISO language codes that specify the character repertoire of your input method. The codes help categorize your input method to the user.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKServer.h`

initWithName:controllerClass:delegateClass:

Creates and returns a server object initialized with the provided parameters.

```
- (id)initWithName:(NSString*)name controllerClass:(Class)controllerClassID
    delegateClass:(Class)delegateClassID
```

Parameters

`name`
The name to initialize the server object with.

`controllerClassID`
The id for the input controller class.

`delegateClassID`
The id for the delegate class.

Return Value
An initialized server object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKServer.h`

Constants

IMKModeDictionary

The input method mode dictionary key.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
extern const NSString* IMKModeDictionary;
```

Constants
`IMKModeDictionary`
The key used to obtain an input method mode dictionary from the input method bundle.

Available in Mac OS X v10.5 and later.

Declared in `IMKServer.h`.

Discussion
For details on the contents of the input mode dictionary, see *Technical Note TN2128 Frequently Asked Text Services Manager (TSM) Questions* located in Technical Notes > Carbon > Events & Other Input.

Declared In
`IMKServer.h`

IMKControllerClass

The input method controller class key.

```
extern const NSString* IMKControllerClass;
```

Constants
`IMKControllerClass`
The key used to find an input method input controller class name from the input method bundle.

Available in Mac OS X v10.5 and later.

Declared in `IMKServer.h`.

Declared In
`IMKServer.h`

IMKDelegateClass

The input method delegate class key.

```
extern const NSString* IMKDelegateClass;
```

Constants
`IMKDelegateClass`
The key used to find an input method delegate class name from the input method bundle.

Available in Mac OS X v10.5 and later.

Declared in `IMKServer.h`.

Declared In
`IMKServer.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Protocols

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKMouseHandling Protocol Reference

Adopted by	IMKInputController
Framework	System/Library/Frameworks/InputMethodKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	InputMethodKit/IMKInputController.h

Overview

The `IMKMouseHandling` protocol defines methods that your input method can implement to handle mouse events.

Tasks

Handling Mouse Events

`- mouseDownOnCharacterIndex:coordinate:withModifier:continueTracking:client:` (page 39)
Handles mouse-down event send to an input method.

`- mouseUpOnCharacterIndex:coordinate:withModifier:client:` (page 41)
Handles a mouse-up event sent to an input method.

`- mouseMovedOnCharacterIndex:coordinate:withModifier:client:` (page 40)
Handles a mouse-moved event sent to an input method.

Instance Methods

mouseDownOnCharacterIndex:coordinate:withModifier:continueTracking:client:

Handles mouse-down event send to an input method.

```
-(BOOL)mouseDownOnCharacterIndex:(NSUInteger)index coordinate:(NSPoint)point
    withModifier:(NSUInteger)flags continueTracking:(BOOL*)keepTracking
    client:(id)sender
```


2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Parameters

index

The index within the sender's text storage where the mouse-down event occurred.

point

The point at which the mouse-down event occurred.

flags

The modifier keys.

keepTracking

Set this parameter to `YES` if you want to receive subsequent mouse-moved and mouse -up events.

sender

The client object.

Return Value
`YES` if handled; otherwise `NO`.

Discussion
Implement this method if your input method handles mouse-down events.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

mouseMovedOnCharacterIndex:coordinate:withModifier:client:

Handles a mouse-moved event sent to an input method.

```
-(BOOL)mouseMovedOnCharacterIndex:(NSUInteger)index coordinate:(NSPoint)point
    withModifier:(NSUInteger)flags client:(id)sender
```

Parameters

index

The index within the sender's text storage where the mouse-moved event occurred.

point

The point at which the mouse-moved event occurred.

flags

The modifier keys.

sender

The client object.

Return Value
`YES` if handled; otherwise `NO`.

Discussion
Implement this method if your input method handles mouse-moved events.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

mouseUpOnCharacterIndex:coordinate:withModifier:client:

Handles a mouse-up event sent to an input method.

```
-(BOOL)mouseUpOnCharacterIndex:(NSUInteger)index coordinate:(NSPoint)point
    withModifier:(NSUInteger)flags client:(id)sender
```

Parameters

index

The index within the sender's text storage where the mouse-up event occurred.

point

The point at which the mouse-up event occurred.

flags

The modifier keys.

sender

The client object.

Return Value

`YES` if handled; otherwise `NO`.

Discussion

Implement this method if your input method handles mouse-up events.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKServerInput Protocol Reference

(informal protocol)

Framework	System/Library/Frameworks/InputMethodKit.framework
Declared in	InputMethodKit/IMKInputController.h

Overview

`IMKServerInput` is an informal protocol that defines methods for receiving text events. This is intentionally not a formal protocol because there are three ways to receive events. An input method chooses one of the following approaches and implements the appropriate methods:

- Key binding. In this approach the system tries to map each key-down event to an action method that the input method has implemented. If successful (action method found), the system calls `didCommandBySelector:client:`. If unsuccessful (action method not found), the system calls `inputText:client:`. For this approach you need to implement `inputText:client:` (page 47) and `didCommandBySelector:client:` (page 45).
- Text data only. In this approach, you opt to receive all key events without the key binding, and then unpack the relevant text data. Key events are broken down into the Unicodes, the key code that generated them, and modifier flags. This data is then sent to the `inputText:key:modifiers:client:` (page 47) method, which you need to implement.
- Handle all events. In this approach, you receive events directly from the Text Services Manager as `NSEvent` objects. You must implement `handleEvent:client:` (page 46) method.

Tasks

Supporting Key Binding

`- inputText:client:` (page 47)
Handles key down events that do not map to an action method.

`- didCommandBySelector:client:` (page 45)
Processes a command generated by user action such as typing certain keys or pressing the mouse button.

Unpacking Text Data

`- inputText:key:modifiers:client:` (page 47)
Receives Unicode, the key code that generated it, and any modifier flags.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Receiving Events Directly from the Text Services Manager

- `handleEvent:client:` (page 46)
  Handles key down and mouse events.

Committing a Composition

- `commitComposition:` (page 45)
  Informs the controller that the composition should be committed.

Getting Input Strings and Candidates

- `composedString:` (page 45)
  Return the current composed string.
- `originalString:` (page 48)
  Return the a string that consists of the precomposed unicode characters.
- `candidates:` (page 44)
  Returns an array of candidates.

Instance Methods

candidates:

Returns an array of candidates.

```
- (NSArray*)candidates:(id)sender
```

Parameters

sender
The client object requesting the candidates.

Return Value
An array of candidates. The returned array should be an autoreleased object.

Discussion
An input method should look up its currently composed string and return a list of candidate strings that that string might map to.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

commitComposition:

Informs the controller that the composition should be committed.

```
- (void)commitComposition:(id)sender
```

Parameters

sender

The client object requesting the input method to commit the composition.

Discussion

If an input method implements this method, it is called when the client wants to end the composition session immediately. A typical response would be to call the `insertText` method of the client and then clean up any per-session buffers and variables. After receiving this message an input method should consider the given composition session finished.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

composedString:

Return the current composed string.

```
- (id)composedString:(id)sender
```

Parameters

sender

The client object requesting the string.

Return Value

The current composed string, which can be an `NSString` or `NSAttributedString` object. The returned object should be an autoreleased object.

Discussion

A composed string refers to the buffer that an input method typically maintains to mirror the text contained in the active inline area. It is called the composed string to reflect the fact that the input method composed the string by converting the characters input by the user. In addition, using the term composed string makes it easier to differentiate between an input method buffer and the text in the active inline area that the user sees.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

didCommandBySelector:client:

Processes a command generated by user action such as typing certain keys or pressing the mouse button.

```
- (BOOL)didCommandBySelector:(SEL)aSelector client:(id)sender
```


2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Parameters

aSelector

The action associated with the key down event. The selector can be an action specified in the input method dictionary of keys and actions (that is, an action specific to the input method) or one of the `NSResponder` action methods such as `insertNewline:` or `deleteBackward:`. By definition such action methods do not return a value.

sender

The client object sending the key down event.

Return Value

`YES` if the command is handled; `NO` if the command is not handled. If not handled, the event passes to the client.

Discussion

This method is called when the system binds a key down event to an action method. If you implement this method you should test if it is appropriate to call the action method before actually calling it, because calling the action method implies that you agree to handle the command. Suppose you have implemented a version of `insertNewline:` that terminates the conversion session and sends the fully converted text to the client. However, if you conversion buffer is empty, you want the application to receive the return key that triggered the call to `insertNewline:`. In that case, when `didCommandBySelector:client:` is called you should test your buffer before calling your implementation of `insertNewline:`. If the buffer is empty, return `NO` to indicate that the return key should be passed on to the application. If the buffer is not empty, call `insertNewline:` and then return `YES` as the result of `didCommandBySelector:client:`.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- inputText:client:` (page 47)

Declared In

`IMKInputController.h`

handleEvent:client:

Handles key down and mouse events.

```
- (BOOL)handleEvent:(NSEvent*)event client:(id)sender
```

Parameters

event

The event to handle.

sender

The client object sending the event.

Return Value

`YES` if the event is handled; otherwise `NO`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

inputText:client:

Handles key down events that do not map to an action method.

```
- (BOOL)inputText:(NSString*)string client:(id)sender
```

Parameters

string

The key down event, which is the text input by the client.

sender

The client object sending the key down events.

Return Value

`YES` if the input is accepted; otherwise `NO`.

Discussion

An input method should implement this method when using key binding (that is, it implements `didCommandBySelector:client:` (page 45)).

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

inputText:key:modifiers:client:

Receives Unicode, the key code that generated it, and any modifier flags.

```
- (BOOL)inputText:(NSString*)string key:(NSInteger)keyCode
    modifiers:(NSUInteger)flags client:(id)sender
```

Parameters

string

The text input by the client.

keyCode

The key code for the associated Unicode.

flags

The modifier flags.

sender

The client object.

Return Value

`YES` if the input is accepted; otherwise `NO`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

originalString:

Return the a string that consists of the precomposed unicode characters.

```
- (NSAttributedString*)originalString:(id)sender
```

Parameters

sender

The client object requesting the original string.

Return Value

The original string of precomposed unicode characters. If an input method stores the original input text, it returns that text. The return value is an attributed string so that the input method can restore changes they made to the font, and other attributes, if necessary. The returned object should be an autoreleased object.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

Constants

Info Dictionary Keys

Constants for keys used to look up information in the info dictionary.

```
extern const NSString *kIMKCommandMenuItemName;
extern const NSString *kIMKCommandClientName;
```

Constants

`kIMKCommandMenuItemName`

Used to look up the `NSMenuItem` object that is passed to menu item actions.

Available in Mac OS X v10.5 and later.

Declared in `IMKInputController.h`.

`kIMKCommandClientName`

Used to look up the client object; the client conforms to the IMKInputText and NSObject protocols.

Available in Mac OS X v10.5 and later.

Declared in `IMKInputController.h`.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKStateSetting Protocol Reference

Adopted by	IMKInputController
Framework	System/Library/Frameworks/InputMethodKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	InputMethodKit/IMKInputController.h

Overview

The `IMKStateSetting` protocol defines methods for setting or accessing values that indicate the state of an input method.

Tasks

Activating and Deactivating the Server

- `activateServer:` (page 50)
  Activates the input method server.
- `deactivateServer:` (page 50)
  Deactivates the input method server.

Showing a Preferences Window

- `showPreferences:` (page 52)
  Displays a preferences window.

Getting the Supported Events

- `recognizedEvents:` (page 51)
  Returns an unsigned integer that contains a union of event masks

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Getting the Mode Dictionary

`- modes:` (page 51)
Returns the modes dictionary associated with the input method.

Getting and Setting Values

`- valueForTag:client:` (page 52)
Returns a value object whose key is the provided tag.

`- setValue:forTag:client:` (page 52)
Set the value for the provided key.

Instance Methods

activateServer:

Activates the input method server.

```
- (void)activateServer:(id)sender
```

Parameters

sender
The object sending the activation message.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- deactivateServer:` (page 50)

Declared In
`IMKInputController.h`

deactivateServer:

Deactivates the input method server.

```
- (void)deactivateServer:(id)sender
```

Parameters

sender
The object sending the deactivation message.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- activateServer:` (page 50)

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
`IMKInputController.h`

modes:

Returns the modes dictionary associated with the input method.

```
- (NSDictionary*)modes:(id)sender
```

Parameters

sender

The client object requesting the modes dictionary.

Return Value
The modes dictionary associated with the input method. The dictionary should be an autoreleased object.

Discussion
Typically a client object calls this method to to build the text input menu. By calling the input method rather than reading the modes from the `Info.plist` file, the input method can dynamically modify the modes supported.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

recognizedEvents:

Returns an unsigned integer that contains a union of event masks

```
- (NSUInteger)recognizedEvents:(id)sender
```

Parameters

sender

The client object requesting the supported events.

Return Value
An unsigned integer that contains a union of event masks (See the `NSEvent.h` header file.

Discussion
A client calls this method to check whether an input method supports an event. The default implementation returns `NSKeyDownMask`. If your input method handles only key down events, the Input Method Kit provides the default mouse handling. The default mouse-down handling behavior is as follows: If there is an active composition area and the user clicks in the text but outside of the composition area, the Input Method Kit sends your input method a `commitComposition:` message. This happens only for input methods that return only the default value—`NSKeyDownMask`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

setValue:forTag:client:

Set the value for the provided key.

```
- (void)setValue:(id)value forTag:(long)tag client:(id)sender
```

Parameters

value

The value, specified as the appropriate object (such as `NSNumber`), to set.

tag

The key whose value you want to set.

sender

The client setting the value.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- valueForTag:client:` (page 52)

Declared In
`IMKInputController.h`

showPreferences:

Displays a preferences window.

```
- (void)showPreferences:(id)sender
```

Parameters

sender

The object sending the message to show the preference window.

Discussion
This method looks for a nib file that contains a window controller class and a preferences utility. If found, it displays the window. To use this method you must create a menu item in your input method menu whose action is `showPreferences:`. When a user selects that item, the Input Method Kit invokes your `showPreferences:` method. The default implementation looks for a nib file named `preferences.nib`. If found, it allocates a window controller class loads the nib file. You can provide a custom window controller class by naming the class in your input method `info.plist` file, providing a key-value pair. The key must be `InputMethodServerPreferencesWindowControllerClass` and the associated value must be the name of your custom class.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

valueForTag:client:

Returns a value object whose key is the provided tag.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

```
- (id)valueForTag:(long)tag client:(id)sender
```

Parameters

tag

The key whose value you want to retrieve.

sender

The client requesting the value.

Return Value

The value object. The returned object should be autoreleased.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- setValue:forTag:client:` (page 52)

Declared In

`IMKInputController.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

IMKTextInput Protocol Reference

Adopted by	IMKInputSession
Framework	System/Library/Frameworks/Carbon.framework/HIToolbox.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	HIToolbox/IMKInputSession.h

Overview

The `IMKTextInput` protocol defines methods for communicating with client input sessions. An input method uses this protocol to send text or obtain information about client text.

Tasks

Working with Text Ranges

- `insertText:replacementRange:` (page 58)
  Sends fully converted text to an input session.
- `setMarkedText:selectionRange:replacementRange:` (page 61)
  Inserts the provided text and marks it to indicate that it is part of an active input session.
- `selectedRange` (page 60)
  Returns the current selection range.
- `markedRange` (page 59)
  Returns the range in the document that is occupied by marked text, that is, the current inline session.
- `attributedSubstringFromRange:` (page 56)
  Returns the attributed string for the provided range of text.

Working with Character Indexes

- `characterIndexForPoint:tracking:inMarkedRange:` (page 58)
  Returns the location in the text document that maps to a global point.
- `attributesForCharacterIndex:lineHeightRectangle:` (page 57)
  Returns a dictionary of text attributes for the text at the provided character index.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Getting Attributes and Length

- `length` (page 59)
  Returns the length of the text document.
- `validAttributesForMarkedText` (page 62)
  Returns an array of names for the attributes supported by the receiver.

Managing Text Input

- `overrideKeyboardWithKeyboardNamed:` (page 60)
  Overrides the current keyboard.
- `selectInputMode:` (page 60)
  Selects a new input mode.
- `supportsUnicode` (page 61)
  Tests to see if the current input session supports Unicode text.
- `bundleIdentifier` (page 57)
  Returns the bundle identifier for the process that the input session is attached to.
- `windowLevel` (page 62)
  Returns the window level for a client window.

Instance Methods

attributedSubstringFromRange:

Returns the attributed string for the provided range of text.

```
- (NSAttributedString*)attributedSubstringFromRange:(NSRange)range
```

Parameters

range
  The range of text, relative to the document, that specifies the string to retrieve.

Return Value
The attributed string. See the `CTStringAttributes.h` header file for the attributes that can be included in this string. If the client does not support the TSMDocumentAccess protocol, the returned string is created from data obtained by sending the client application a `kEventTextInputGetSelectedText` Carbon event. The returned `NSAttributedString` object is an autoreleased object.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- attributesForCharacterIndex:lineHeightRectangle:` (page 57)

Declared In
`IMKInputSession.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

attributesForCharacterIndex:lineHeightRectangle:

Returns a dictionary of text attributes for the text at the provided character index.

```
- (NSDictionary*)attributesForCharacterIndex:(NSUInteger)index
    lineHeightRectangle:(NSRect*)lineRect
```

Parameters

index

The character index whose attributes you want to retrieve. The index is relative to the inline session. Note that if there is no inline session the value of index should be `0`, which indicates that the information should be taken from the current selection.

lineRect

On return, a rectangle that frames a one-pixel wide rectangle with the height of the line. This rectangle is oriented the same way the line is oriented.

Return Value
A dictionary hat contains the text attributes for the text at the provided character index. The returned `NSDictionary` object is an autoreleased object. The attributes include the `CTFontRef` for the text at that index, and the text orientation. The text orientation is indicated by an `NSNumber` whose value is 0 if the text is vertically oriented and 1 if the text is horizontally oriented. The key for this value is `IMKTextOrientationKey`.

Discussion
An input method calls this method to place a candidate window on screen.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- characterIndexForPoint:tracking:inMarkedRange:` (page 58)

Declared In
`IMKInputSession.h`

bundleIdentifier

Returns the bundle identifier for the process that the input session is attached to.

```
-(NSString*)bundleIdentifier
```

Return Value
The bundle identifier for the process that the input session is attached to. The returned `NSString` is an autoreleased object.

Discussion
Many input methods need to be able to identify the process that input sessions belong to. This method provides that service.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputSession.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

characterIndexForPoint:tracking:inMarkedRange:

Returns the location in the text document that maps to a global point.

```
- (NSInteger)characterIndexForPoint:(NSPoint)point
    tracking:(IMKLocationToOffsetMappingMode)mappingMode
    inMarkedRange:(BOOL*)inMarkedRange
```

Parameters

point

The point to map. This is a global point, typically from a mouse down operation.

mappingMode

The mapping mode. If the input method is tracking the mouse, the mode should be `kIMKMouseTrackingMode`. If the input method simply wants to map a screen position to an offset, than set the mode to k`IMKNearestBoundaryMode`.

inMarkedRange

On return, if the point is inside the text body and inside the marked range, this parameter is set to `YES`. Otherwise, if the point is outside the marked range, then on return this parameter is set to `NO`.

Return Value
The location in the text document.

Discussion
If the input method is tracking the mouse, the application should highlight to the active inline area.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- attributesForCharacterIndex:lineHeightRectangle:` (page 57)

Declared In
`IMKInputSession.h`

insertText:replacementRange:

Sends fully converted text to an input session.

```
- (void)insertText:(id)string replacementRange:(NSRange)replacementRange
```

Parameters

string

The converted text.

replacementRange

The replacement range. This parameter allows input methods to insert text at a location other than the current selection. If you use it, the replacement range should be relative to the beginning of the client document. If the string should be inserted at the current selection specify a replacement range with a location and length of `NSNotFound`. If the client does not support the `TSMDocumentAccess` protocol, this method ignores the replacement range string.

Discussion
When an input method finishes a conversion it calls this method and passes the finished text as an `NSString` or `NSAttributedString` object.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputSession.h`

length

Returns the length of the text document.

```
- (NSInteger)length
```

Return Value
The text document length. If the client does not support the `TSMDocumentAccess` protocol the returned value is `NSNotFound`.

Discussion
This method can be computationally expense depending on how the client stores text. For that reason, you should avoid calling this method frequently.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputSession.h`

markedRange

Returns the range in the document that is occupied by marked text, that is, the current inline session.

```
- (NSRange)markedRange
```

Return Value
The range of the marked text.

Discussion
While an input method is converting text input and sending it to the client, the client maintains a range of text that is marked. The marked text is underlined in certain ways to indicate to the user that their keystrokes are in the process of being converted by an input method and the conversion has not yet been finalized. The input method uses this method to request that range.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setMarkedText:selectionRange:replacementRange:` (page 61)

Declared In
`IMKInputSession.h`

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

overrideKeyboardWithKeyboardNamed:

Overrides the current keyboard.

```
-(void)overrideKeyboardWithKeyboardNamed:(NSString*)keyboardUniqueName
```

Parameters

keyboardUniqueName

A unique keyboard name.

Discussion

The client tries to locate a keyboard layout with that name in the input method bundle. If a layout is found it is passed to the client who then tells the text service manager to use that layout for keyboard events. Input methods should call the method each time they are activated. If an input method uses a system keyboard to override the current keyboard, they are responsible for determining the unique name of the keyboard. Typically this name a DNS type name such as: `com.apple.<some name>`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputSession.h`

selectedRange

Returns the current selection range.

```
- (NSRange)selectedRange
```

Return Value

The current selection range, relative to the client document. If the client does not support the `TSMDocumentAccess` protocol the returned range has a location value of `NSNotFound` and a length of `NSNotFound`.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- setMarkedText:selectionRange:replacementRange:` (page 61)

Declared In

`IMKInputSession.h`

selectInputMode:

Selects a new input mode.

```
-(void)selectInputMode:(NSString*)modeIdentifier
```

Parameters

modeIdentifier

An `NSString` object with a DNS format, such as: `com.<company name> .inputmethod.<some name>`. The identifier should match one of the keys in the component input mode dictionary.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Discussion
This method allows an input method to change its mode directly.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputSession.h`

setMarkedText:selectionRange:replacementRange:

Inserts the provided text and marks it to indicate that it is part of an active input session.

```
- (void)setMarkedText:(id)string selectionRange:(NSRange)selectionRange
    replacementRange:(NSRange)replacementRange
```

Parameters

`string`

An `NSString` or an `NSAttributedString` object. Passing an `NSString` object produces default marking. On a 72 dpi screen the default marking is a 2 pixel black underline for the entire string.

`selectionRange`

The selection range, which is relative to the `string` parameter. For example, if the string contains the these characters: "INPUT" and the range is (5,0), the selection is set immediately after the "T."

`replacementRange`

A range that specifies the location in the client document where the marked text should be placed. If the marked text should be placed at the current cursor location `replacementRange` equals `NSNotFound`. If not `NSNotFound`, the replacement range is relative to the client document (`0` is the beginning of the document) and NOT the string parameter. Currently, the replacement range is sent to the client via the kEventParamTextInputSendReplaceRange Carbon Event parameter. For this parameter to be used the client must support the TSMDocumentAccess protocol. If a client does not support the TSMDocumentAccess protocol the `replacementRange` parameter is ignored.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `markedRange` (page 59)
- `selectedRange` (page 60)

Declared In
`IMKInputSession.h`

supportsUnicode

Tests to see if the current input session supports Unicode text.

```
-(BOOL)supportsUnicode
```

Return Value
`YES` if the current input session supports Unicode text; otherwise `NO`.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Discussion
Input methods that restrict the character codes sent to a client if that client does not support Unicode text should call this method to learn whether a given input session supports Unicode text.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputSession.h`

validAttributesForMarkedText

Returns an array of names for the attributes supported by the receiver.

```
- (NSArray*)validAttributesForMarkedText
```

Return Value
An array of attribute names. The returned `NSArray` should not be released unless it is first retained.

Discussion
Input methods should restrict the attributes used to create attributed strings to the attributes in this array.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- attributesForCharacterIndex:lineHeightRectangle:` (page 57)
`- markedRange` (page 59)

Declared In
`IMKInputSession.h`

windowLevel

Returns the window level for a client window.

```
-(CGWindowLevel)windowLevel
```

Return Value
The window level for a client window.

Discussion
Call this to determine the client window level. Internally, the `IMKCandidates` class uses this method to determine the correct level for candidate display. Therefore, input methods that use `IMKCandidates` have no reason to call this method. However, many input method developers build their own candidate display system, and this method is provided for their use. In order to display a candidate window at the correct level, use this method to obtain the client window level, increment the returned level, and then apply that level to any candidate windows.

Availability
Available in Mac OS X v10.5 and later.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Declared In
IMKInputSession.h

Constants

IMKLocationToOffsetMappingMode

List modes for mapping a screen location to a text offset.

```
enum {   kIMKMouseTrackingMode = (1<<0),   kIMKNearestBoundaryMode = (1<<1) };typedef
 NSInteger IMKLocationToOffsetMappingMode;
```

Constants

`kIMKMouseTrackingMode`
Mouse tracking mode. When tracking, a coordinate does not change the offset until the coordinate is completely past a given character.

Available in Mac OS X v10.5 and later.

Declared in `IMKInputSession.h`.

`kIMKNearestBoundaryMode`
Nearest boundary mode. When trying to place an insertion point, a coordinate is mapped to the nearest character boundary.

Available in Mac OS X v10.5 and later.

Declared in `IMKInputSession.h`.

Declared In
IMKInputSession.h

IMKTextOrientationName

The text orientation name key.

```
extern const NSString* IMKTextOrientationName;
```

Constants

`IMKTextOrientationName`
The key used to find the client text orientation value, which is part of the dictionary returned by the method `attributesForCharacterIndex:lineHeightRectangle:` (page 57). The associated value is an `NSNumber` object that represents a `BOOL` value. A value of `1` means the client text uses a horizontal layout. A value of `0` means that the client text uses a vertical layout.

Available in Mac OS X v10.5 and later.

Declared in `IMKInputSession.h`.

Declared In
IMKInputSession.h

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *Input Method Kit Framework Reference*.

Date	Notes
2007-06-06	New document that describes the API for building input methods for Chinese, Japanese, and other languages.

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Document Revision History

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

Index

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

K

L

M

O

P

R

S

U

V

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.

W

2007-06-06 | © 2004, 2007 Apple Inc. All Rights Reserved.