NSInputServiceProvider Protocol Reference

Cocoa > Events & Other Input

2007-04-02

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

Simultaneously published in the United States and Canada.

Contents

NSInputServiceProvider Protocol Reference

Adopted by	NSInputServer
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Text Input Management
Declared in	NSInputServer.h

Overview

The NSInputServiceProvider protocol embodies most of the functionality of NSInputServer.

There are two ways you might use this protocol:

- You can subclass NSInputServer and create an instance of your subclass. Your subclass must override most or all of the NSInputServiceProvider protocol methods.
- You can create an NSInputServer object and designate a delegate. The delegate must implement the NSInputServiceProvider protocol.

All messages in this protocol are sent by the client text view except `insertText:client:` (page 9) and `doCommandBySelector:client:` (page 7), which are sent by "NSInputManager".

Tasks

Getting Input Service Provider Information

`- canBeDisabled` (page 7)
Returns `YES` if the receiver can be disabled when the sender is not a text view, `NO`

Handling Events

`- wantsToDelayTextChangeNotifications` (page 11)
A `YES` return value tells the client that only a call to its `insertText:client:` (page 9) method constitutes a modification to its text storage.

- `wantsToHandleMouseEvents` (page 11)
  Returns `YES` if the client should forward all mouse events within the text view to the input server.
- `wantsToInterpretAllKeystrokes` (page 12)
  Returns `YES` if the server wants all keystrokes to be sent to it as characters.
- `doCommandBySelector:client:` (page 7)
  Handle the command identified by *`aSelector`*.
- `insertText:client:` (page 9)
  Interpret the characters in *`aString`*, which is actually always an `NSString`.
- `activeConversationChanged:toNewConversation:` (page 6)
  Keyboard focus just switched from another text view to this one.
- `activeConversationWillChange:fromOldConversation:` (page 7)
  Keyboard focus is about to move away from this text view.
- `inputClientBecomeActive:` (page 8)
  The client, *`sender`*, has become active.
- `inputClientEnabled:` (page 9)
  A text view in the client, *`sender`*, has become the key-receiving first responder.
- `inputClientDisabled:` (page 8)
  A text view in the client, *`sender`*, has ceased to be the key-receiving first responder.
- `inputClientResignActive:` (page 9)
  The client, *`sender`*, is about to become inactive.
- `markedTextAbandoned:` (page 10)
  Abandon any marked text state that may be in process.
- `markedTextSelectionChanged:client:` (page 10)

- `terminate:` (page 10)
  The client application is quitting.

Instance Methods

activeConversationChanged:toNewConversation:

Keyboard focus just switched from another text view to this one.

```
- (void)activeConversationChanged:(id)sender
    toNewConversation:(NSInteger)newConversation
```

Discussion
This is called only when switching within the same application. *`sender`* can be cast to NSTextInput.

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

See Also
- `activeConversationWillChange:fromOldConversation:` (page 7)
- `conversationIdentifier` (NSTextInput)

Declared In
NSInputServer.h

activeConversationWillChange:fromOldConversation:

Keyboard focus is about to move away from this text view.

```
- (void)activeConversationWillChange:(id)sender
    fromOldConversation:(NSInteger)oldConversation
```

Discussion
This is called only when switching within the same application. *sender* can be cast to NSTextInput.

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

See Also
- activeConversationChanged:toNewConversation: (page 6)
- conversationIdentifier (NSTextInput)

Declared In
NSInputServer.h

canBeDisabled

Returns YES if the receiver can be disabled when the sender is not a text view, NO

```
- (BOOL)canBeDisabled
```

Discussion
otherwise.

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

Declared In
NSInputServer.h

doCommandBySelector:client:

Handle the command identified by *aSelector*.

```
- (void)doCommandBySelector:(SEL)aSelector client:(id)sender
```

Discussion
The command can be from the set of NSResponder action methods or from the set of selector values in the DefaultKeyBindings dictionary referenced in the input server's "Info" file. *sender* can be cast to NSTextInput.

If you are subclassing NSInputServer, there is no need to override this method in the subclass. All you have to do is implement in the subclass the command methods you want to handle. If you do need to override this method, then you must call super for commands not handled.

If your NSInputServer uses a delegate, the delegate's implementation of this method must call `[sender doCommandBySelector:aSelector]` for commands it does not handle.

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

See Also
`- doCommandBySelector:` (NSTextInput)

Declared In
`NSInputServer.h`

inputClientBecomeActive:

The client, *sender*, has become active.

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

Discussion
This is called when the client application starts up and whenever it becomes active after being inactive. *sender* can be cast to NSTextInput.

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

See Also
- `inputClientEnabled:` (page 9)
- `inputClientResignActive:` (page 9)

Declared In
`NSInputServer.h`

inputClientDisabled:

A text view in the client, *sender*, has ceased to be the key-receiving first responder.

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

Discussion
`inputClientResignActive:` (page 9) may also be called just after this is called. *sender* can be cast to NSTextInput.

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

See Also
- `inputClientEnabled:` (page 9)
- `inputClientResignActive:` (page 9)

Declared In
`NSInputServer.h`

inputClientEnabled:

A text view in the client, *sender*, has become the key-receiving first responder.

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

Discussion
This is called the first time any text view becomes enabled after client application activation and again whenever focus switches to a text view. `inputClientBecomeActive:` (page 8) may have been called just before this is called. *sender* can be cast to NSTextInput.

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

See Also
- `inputClientBecomeActive:` (page 8)
- `inputClientDisabled:` (page 8)

Declared In
`NSInputServer.h`

inputClientResignActive:

The client, *sender*, is about to become inactive.

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

Discussion
This is called when the client application quits and whenever it is deactivated. *sender* can be cast to NSTextInput.

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

See Also
- `inputClientBecomeActive:` (page 8)
- `inputClientDisabled:` (page 8)
- `terminate:` (page 10)

Declared In
`NSInputServer.h`

insertText:client:

Interpret the characters in *aString*, which is actually always an `NSString`.

```
- (void)insertText:(id)aString client:(id)sender
```

Discussion
Here is where you do the interpreting of keyboard input. If your server's interpretation is disabled or the characters in *aString* are not of interest to the server, you can simply pass *aString* along to the sender's `insertText:` method. *sender* can be cast to NSTextInput.

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

See Also
- `insertText:` (NSTextInput)

Declared In
`NSInputServer.h`

markedTextAbandoned:

Abandon any marked text state that may be in process.

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

Discussion
This can happen if the user clicks the mouse outside of the marked text area or if the window containing the text view closes. The client can do what it wants with the marked text. `NSTextView` leaves it as inserted text. *`sender`* can be cast to NSTextInput.

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

See Also
- `markedTextSelectionChanged:client:` (page 10)
- `markedTextAbandoned:` (NSInputManager)

Declared In
`NSInputServer.h`

markedTextSelectionChanged:client:

```
- (void)markedTextSelectionChanged:(NSRange)newSelection client:(id)sender
```

Discussion
The user selected a portion of the marked text or clicked at the beginning or end of marked text or somewhere in between. *`sender`* can be cast to NSTextInput.

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

See Also
- `markedTextAbandoned:` (page 10)
- `markedTextSelectionChanged:client:` (NSInputManager)

Declared In
`NSInputServer.h`

terminate:

The client application is quitting.

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

Discussion
This is called after `inputClientResignActive:` (page 9). *sender* can be cast to NSTextInput.

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

See Also
`- inputClientResignActive:` (page 9)

Declared In
`NSInputServer.h`

wantsToDelayTextChangeNotifications

A `YES` return value tells the client that only a call to its `insertText:client:` (page 9) method constitutes a modification to its text storage.

```
- (BOOL)wantsToDelayTextChangeNotifications
```

Discussion
A `NO` return value tells the client that all text given to it, whether marked text or not, should constitute a modification to its text storage. A `YES` return value tells the client that only unmarked text given to it should constitute a modification to its text storage. The client may for example want to filter all text that is part of a modification but leave marked text unfiltered.

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

See Also
`- wantsToDelayTextChangeNotifications` (NSInputManager)

Declared In
`NSInputServer.h`

wantsToHandleMouseEvents

Returns `YES` if the client should forward all mouse events within the text view to the input server.

```
- (BOOL)wantsToHandleMouseEvents
```

Discussion
If the server needs to implement the NSInputServerMouseTracker protocol, return `YES`.

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

See Also
`- wantsToHandleMouseEvents` (NSInputManager)

Declared In
`NSInputServer.h`

wantsToInterpretAllKeystrokes

Returns `YES` if the server wants all keystrokes to be sent to it as characters.

```
- (BOOL)wantsToInterpretAllKeystrokes
```

Discussion
If this method returns `NO`, control key combinations and function keys (the arrow keys, PageDown, F5, and so on) are delivered to the input server via the key binding mechanism and `doCommandBySelector:client:` (page 7).

The Unicode values for the characters representing keyboard function keys (the arrow keys, PageDown, F5, and so on) names like `NSUpArrowFunctionKey`, and are documented in NSEvent. Control-key combinations are the usual ASCII control character codes.

For more information on key bindings, see "About Key Bindings".

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

See Also
`- wantsToInterpretAllKeystrokes` (NSInputManager)

Declared In
`NSInputServer.h`

Document Revision History

This table describes the changes to *NSInputServiceProvider Protocol Reference*.

Date	Notes
2007-04-02	Made editorial improvements.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index