
NSAppleEventManager Class Reference

Interapplication Communication



2006-05-23



Apple Inc.
© 2006 Apple Computer, 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, AppleScript, Carbon, Cocoa, Mac, and Mac OS 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

NSAppleEventManager Class Reference 5

Overview	5
Tasks	6
Getting an Event Manager	6
Working with Event Handlers	6
Working with Events	6
Suspending and Resuming Apple Events	6
Class Methods	7
sharedAppleEventManager	7
Instance Methods	7
appleEventForSuspensionID:	7
currentAppleEvent	8
currentReplyAppleEvent	8
dispatchRawAppleEvent:withRawReply:handlerRefCon:	9
removeEventHandlerForEventClass:andEventID:	9
replyAppleEventForSuspensionID:	9
resumeWithSuspensionID:	10
setCurrentAppleEventAndReplyEventWithSuspensionID:	10
setEventHandler:andSelector:forEventClass:andEventID:	11
suspendCurrentAppleEvent	11
Constants	12
NSAppleEventTimeouts	12
Notifications	12
NSAppleEventManagerWillProcessFirstEventNotification	12

Document Revision History 13

Index 15

NSAppleEventManager Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Cocoa Scripting Guide
Declared in	NSAppleEventManager.h
Related sample code	CoreRecipes SampleRaster SimpleScriptingPlugin Sketch-112

Overview

Provides a mechanism for registering handler routines for specific types of Apple events and dispatching events to those handlers.

Cocoa provides built-in scriptability support that uses scriptability information supplied by an application to automatically convert Apple events into script command objects that perform the desired operation. However, some applications may want to perform more basic Apple event handling, in which an application registers handlers for the Apple events it can process, then calls on the Apple Event Manager to dispatch received Apple events to the appropriate handler. `NSAppleEventManager` supports these mechanisms by providing methods to register and remove handlers and to dispatch Apple events to the appropriate handler, if one exists. For related information, see “How Cocoa Applications Handle Apple Events.”

Each application has at most one instance of `NSAppleEventManager`. To obtain a reference to it, you call the class method `sharedAppleEventManager` (page 7), which creates the instance if it doesn't already exist.

For information about the Apple Event Manager, see *Apple Event Manager Reference* and *Apple Events Programming Guide*.

Tasks

Getting an Event Manager

- + [sharedAppleEventManager](#) (page 7)
Returns the single instance of `NSAppleEventManager`, creating it first if it doesn't exist.

Working with Event Handlers

- [removeEventHandlerForEventClass:andEventID:](#) (page 9)
If an Apple event handler has been registered for the event specified by *eventClass* and *eventID*, removes it.
- [setEventHandler:andSelector:forEventClass:andEventID:](#) (page 11)
Registers the Apple event handler specified by *handler* for the event specified by *eventClass* and *eventID*.

Working with Events

- [dispatchRawAppleEvent:withRawReply:handlerRefCon:](#) (page 9)
Causes the Apple event specified by *theAppleEvent* to be dispatched to the appropriate Apple event handler, if one has been registered by calling [setEventHandler:andSelector:forEventClass:andEventID:](#) (page 11).

Suspending and Resuming Apple Events

- [appleEventForSuspensionID:](#) (page 7)
Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), returns the descriptor for the event whose handling was suspended.
- [currentAppleEvent](#) (page 8)
Returns the descriptor for *currentAppleEvent* if an Apple event is being handled on the current thread.
- [currentReplyAppleEvent](#) (page 8)
Returns the corresponding reply event descriptor if an Apple event is being handled on the current thread.
- [replyAppleEventForSuspensionID:](#) (page 9)
Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), returns the corresponding reply event descriptor.
- [resumeWithSuspensionID:](#) (page 10)
Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), signal that handling of the suspended event may now continue.
- [setCurrentAppleEventAndReplyEventWithSuspensionID:](#) (page 10)
Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), sets the values that will be returned by subsequent invocations of [currentAppleEvent](#) (page 11).

- 8) and [currentReplyAppleEvent](#) (page 8) to be the event whose handling was suspended and its corresponding reply event, respectively.
- [suspendCurrentAppleEvent](#) (page 11)
Suspends the handling of the current event and returns an ID that must be used to resume the handling of the event if an Apple event is being handled on the current thread.

Class Methods

sharedAppleEventManager

Returns the single instance of `NSAppleEventManager`, creating it first if it doesn't exist.

```
+ (NSAppleEventManager *)sharedAppleEventManager
```

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CoreRecipes
SampleRaster
SimpleScriptingPlugin
Sketch-112

Declared In

`NSAppleEventManager.h`

Instance Methods

appleEventForSuspensionID:

Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), returns the descriptor for the event whose handling was suspended.

```
- (NSAppleEventDescriptor *)appleEventForSuspensionID:(NSAppleEventManagerSuspensionID)suspensionID
```

Discussion

The effects of mutating or retaining the returned descriptor are undefined, although it may be copied. `appleEventForSuspensionID:` may be invoked in any thread, not just the one in which the corresponding invocation of `suspendCurrentAppleEvent` occurred.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [currentAppleEvent](#) (page 8)
- [currentReplyAppleEvent](#) (page 8)

Declared In

NSAppleEventManager.h

currentAppleEvent

Returns the descriptor for *currentAppleEvent* if an Apple event is being handled on the current thread.

- (NSAppleEventDescriptor *)currentAppleEvent

Discussion

An Apple event is being handled on the current thread if a handler that was registered with [setEventHandler:andSelector:forEventClass:andEventID:](#) (page 11) is being messaged at this instant or [setCurrentAppleEventAndReplyEventWithSuspensionID:](#) (page 10) has just been invoked. Returns *nil* otherwise. The effects of mutating or retaining the returned descriptor are undefined, although it may be copied.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [currentReplyAppleEvent](#) (page 8)

Related Sample Code

SampleRaster

Declared In

NSAppleEventManager.h

currentReplyAppleEvent

Returns the corresponding reply event descriptor if an Apple event is being handled on the current thread.

- (NSAppleEventDescriptor *)currentReplyAppleEvent

Discussion

An Apple event is being handled on the current thread if [currentAppleEvent](#) (page 8) does not return *nil*. Returns *nil* otherwise. This descriptor, including any mutations, will be returned to the sender of the current event when all handling of the event has been completed, if the sender has requested a reply. The effects of retaining the descriptor are undefined; it may be copied, but mutations of the copy are not returned to the sender of the current event.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [setCurrentAppleEventAndReplyEventWithSuspensionID:](#) (page 10)

Related Sample Code

Sketch-112

Declared In

NSAppleEventManager.h

dispatchRawAppleEvent:withRawReply:handlerRefCon:

Causes the Apple event specified by *theAppleEvent* to be dispatched to the appropriate Apple event handler, if one has been registered by calling [setEventHandler:andSelector:forEventClass:andEventID:](#) (page 11).

```
- (OSErr)dispatchRawAppleEvent:(const AppleEvent *)theAppleEvent
    withRawReply:(AppleEvent *)theReply handlerRefCon:(UInt32)handlerRefcon
```

Discussion

The *theReply* parameter always specifies a reply Apple event, never `nil`. However, the handler should not fill out the reply if the descriptor type for the reply event is `typeNull`, indicating the sender does not want a reply.

The *handlerRefcon* parameter provides 4 bytes of data to the handler; a common use for this parameter is to pass a pointer to additional data.

This method is primarily intended for Cocoa's internal use. Note that *dispatching* an event means routing an event to an appropriate handler in the current application. You cannot use this method to *send* an event to other applications.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventManager.h

removeEventHandlerForEventClass:andEventID:

If an Apple event handler has been registered for the event specified by *eventClass* and *eventID*, removes it.

```
- (void)removeEventHandlerForEventClass:(AEEEventClass)eventClass
    andEventID:(AEEEventID)eventID
```

Discussion

Otherwise does nothing.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setEventHandler:andSelector:forEventClass:andEventID:](#) (page 11)

Declared In

NSAppleEventManager.h

replyAppleEventForSuspensionID:

Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), returns the corresponding reply event descriptor.

```
- (NSAppleEventDescriptor
    *)replyAppleEventForSuspensionID:(NSAppleEventManagerSuspensionID)suspensionID
```

Discussion

This descriptor, including any mutations, will be returned to the sender of the suspended event when handling of the event is resumed, if the sender has requested a reply. The effects of retaining the descriptor are undefined; it may be copied, but mutations of the copy are returned to the sender of the suspended event. `replyAppleEventForSuspensionID:` may be invoked in any thread, not just the one in which the corresponding invocation of `suspendCurrentAppleEvent` occurred.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [appleEventForSuspensionID:](#) (page 7)
- [currentAppleEvent](#) (page 8)
- [currentReplyAppleEvent](#) (page 8)
- [setCurrentAppleEventAndReplyEventWithSuspensionID:](#) (page 10)

Declared In

NSAppleEventManager.h

resumeWithSuspensionID:

Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), signal that handling of the suspended event may now continue.

```
- (void)resumeWithSuspensionID:(NSAppleEventManagerSuspensionID)suspensionID
```

Discussion

This may result in the immediate sending of the reply event to the sender of the suspended event, if the sender has requested a reply. If *suspensionID* has been used in a previous invocation of [setCurrentAppleEventAndReplyEventWithSuspensionID:](#) (page 10) the effects of that invocation are completely undone. Redundant invocations of `resumeWithSuspensionID:` are ignored. Subsequent invocations of other NSAppleEventManager methods using the same suspension ID are invalid. `resumeWithSuspensionID:` may be invoked in any thread, not just the one in which the corresponding invocation of `suspendCurrentAppleEvent` occurred.

Availability

Available in Mac OS X v10.3 and later.

Declared In

NSAppleEventManager.h

setCurrentAppleEventAndReplyEventWithSuspensionID:

Given a nonzero *suspensionID* returned by an invocation of [suspendCurrentAppleEvent](#) (page 11), sets the values that will be returned by subsequent invocations of [currentAppleEvent](#) (page 8) and [currentReplyAppleEvent](#) (page 8) to be the event whose handling was suspended and its corresponding reply event, respectively.

```
- (void)setCurrentAppleEventAndReplyEventWithSuspensionID:(NSAppleEventManagerSuspensionID)suspensionID
```

Discussion

Redundant invocations of `setCurrentAppleEventAndReplyEventWithSuspensionID:` are ignored.

Availability

Available in Mac OS X v10.3 and later.

Declared In

NSAppleEventManager.h

setEventHandler:andSelector:forEventClass:andEventID:

Registers the Apple event handler specified by *handler* for the event specified by *eventClass* and *eventID*.

```
- (void)setEventHandler:(id)handler andSelector:(SEL)handleEventSelector
    forEventClass:(AEEEventClass)eventClass andEventID:(AEEEventID)eventID
```

Discussion

If an event handler is already registered for the specified event class and event ID, removes it. The signature for *handler* should match the following:

```
- (void)handleAppleEvent:(NSAppleEventDescriptor *)event withReplyEvent:
(NSAppleEventDescriptor *)replyEvent;
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [removeEventHandlerForEventClass:andEventID:](#) (page 9)

Related Sample Code

CoreRecipes

SimpleScriptingPlugin

Declared In

NSAppleEventManager.h

suspendCurrentAppleEvent

Suspends the handling of the current event and returns an ID that must be used to resume the handling of the event if an Apple event is being handled on the current thread.

```
- (NSAppleEventManagerSuspensionID)suspendCurrentAppleEvent
```

Discussion

An Apple event is being handled on the current thread if [currentAppleEvent](#) (page 8) does not return `nil`. Returns zero otherwise. The suspended event is no longer the current event after this method returns.

Availability

Available in Mac OS X v10.3 and later.

See Also

- [currentReplyAppleEvent](#) (page 8)

- [resumeWithSuspensionID:](#) (page 10)

Declared In

NSAppleEventManager.h

Constants

NSAppleEvent Timeouts

The following constants should not be used and may eventually be removed.

```
extern const double NSAppleEventTimeoutDefault;
extern const double NSAppleEventTimeoutNone;
```

Constants

NSAppleEventTimeoutDefault

Specifies that an event-processing operation should continue until a timeout occurs based on a value determined by the Apple Event Manager (about 1 minute). Not currently used by applications.

Available in Mac OS X v10.0 and later.

Declared in NSAppleEventManager.h.

NSAppleEventTimeoutNone

Specifies that the application is willing to wait indefinitely for the current operation to complete. Not currently used by applications.

Available in Mac OS X v10.0 and later.

Declared in NSAppleEventManager.h.

Declared In

NSAppleEventManager.h

Notifications

NSAppleEventManagerWillProcessFirstEventNotification

Posted by NSAppleEventManager before it first dispatches an Apple event. Your application can use this notification to avoid registering any Apple event handlers until the first time at which they may be needed. The notification object is the NSAppleEventManager. This notification does not contain a *userInfo* dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventManager.h

Document Revision History

This table describes the changes to *NSAppleEventManager Class Reference*.

Date	Notes
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

appleEventForSuspensionID: **instance method** [7](#)

C

currentAppleEvent **instance method** [8](#)
currentReplyAppleEvent **instance method** [8](#)

D

dispatchRawAppleEvent:withRawReply:handlerRefCon:
instance method [9](#)

N

NSAppleEvent Timeouts [12](#)
NSAppleEventManagerWillProcessFirstEvent-
Notification **notification** [12](#)
NSAppleEventTimeOutDefault **constant** [12](#)
NSAppleEventTimeOutNone **constant** [12](#)

R

removeEventHandlerForEventClass:andEventID:
instance method [9](#)
replyAppleEventForSuspensionID: **instance method**
[9](#)
resumeWithSuspensionID: **instance method** [10](#)

S

setCurrentAppleEventAndReplyEventWithSuspensionID:
instance method [10](#)
setEventHandler:andSelector:forEventClass:
andEventID: **instance method** [11](#)
sharedAppleEventManager **class method** [7](#)
suspendCurrentAppleEvent **instance method** [11](#)