
NSAppleEventDescriptor Class Reference

Interapplication Communication



2007-04-10



Apple Inc.
© 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, 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

NSAppleEventDescriptor Class Reference 5

Overview	5
Adopted Protocols	6
Tasks	6
Creating and Initializing Descriptors	6
Getting Information About a Descriptor	7
Working With List Descriptors	8
Working With Record Descriptors	8
Working With Apple Event Descriptors	8
Class Methods	9
appleEventWithEventClass:eventID:targetDescriptor:returnID:transactionID:	9
descriptorWithBoolean:	10
descriptorWithDescriptorType:bytes:length:	10
descriptorWithDescriptorType:data:	11
descriptorWithEnumCode:	11
descriptorWithInt32:	12
descriptorWithString:	12
descriptorWithTypeCode:	13
listDescriptor	13
nullDescriptor	13
recordDescriptor	14
Instance Methods	14
aeDesc	14
attributeDescriptorForKeyword:	15
booleanValue	15
coerceToDescriptorType:	15
data	16
descriptorAtIndex:	16
descriptorForKeyword:	17
descriptorType	17
enumCodeValue	17
eventClass	18
eventID	18
initListDescriptor	18
initRecordDescriptor	19
initWithAEDescNoCopy:	19
initWithDescriptorType:bytes:length:	20
initWithDescriptorType:data:	20
initWithEventClass:eventID:targetDescriptor:returnID:transactionID:	21
insertDescriptor:atIndex:	21
int32Value	22

keywordForDescriptorAtIndex: 22
numberOfItems 23
paramDescriptorForKeyword: 23
removeDescriptorAtIndex: 24
removeDescriptorWithKeyword: 24
removeParamDescriptorWithKeyword: 24
returnID 25
setAttributeDescriptor:forKeyword: 25
setDescriptor:forKeyword: 26
setParamDescriptor:forKeyword: 26
stringValue 27
transactionID 27
typeCodeValue 27

Document Revision History 29

Index 31

NSAppleEventDescriptor Class Reference

Inherits from	NSObject
Conforms to	NSCopying 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	NSAppleEventDescriptor.h
Related sample code	Apply Firmware Password AttachAScript CoreRecipes SimpleScriptingPlugin Sketch+Accessibility

Overview

An instance of `NSAppleEventDescriptor` represents a descriptor—the basic building block for Apple events. This class is a wrapper for the underlying Apple event descriptor data type, `AEDesc`. Scriptable Cocoa applications frequently work with instances of `NSAppleEventDescriptor`, but should rarely need to work directly with the `AEDesc` data structure.

A *descriptor* is a data structure that stores data and an accompanying four-character code. A descriptor can store a value, or it can store a list of other descriptors (which may also be lists). All the information in an Apple event is stored in descriptors and lists of descriptors, and every Apple event is itself a descriptor list that matches certain criteria.

Important: An instance of `NSAppleEventDescriptor` can represent any kind of descriptor, from a simple value descriptor, to a descriptor list, to a full-fledged Apple event.

Descriptors can be used to build arbitrarily complex containers, so that one Apple event can represent a script statement such as `tell application "TextEdit" to get word 3 of paragraph 6 of document 3`.

In working with Apple event descriptors, it can be useful to understand some of the underlying data types. You'll find terms such as descriptor, descriptor list, Apple event record, and Apple event defined in *Building an Apple Event in Apple Events Programming Guide*. You'll also find information on the four-character codes

used to identify information within a descriptor. Apple event data types are defined in *Apple Event Manager Reference*. The values of many four-character codes used by Apple (and in some cases reused by developers) can be found in [AppleScript Terminology and Apple Event Codes](#).

The most common reason to construct an Apple event with an instance of `NSAppleEventDescriptor` is to supply information in a return Apple event. The most common situation where you might need to extract information from an Apple event (as an instance of `NSAppleEventDescriptor`) is when an Apple event handler installed by your application is invoked, as described in “Installing an Apple Event Handler” in *How Cocoa Applications Handle Apple Events*. In addition, if you execute an AppleScript script using the `NSAppleScript` class, you get an instance of `NSAppleEventDescriptor` as the return value, from which you can extract any required information.

When you work with an instance of `NSAppleEventDescriptor`, you can access the underlying descriptor directly, if necessary, with the `aeDesc` (page 14) method. Other methods, including `descriptorWithDescriptorType:bytes:length:` (page 10) make it possible to create and initialize instances of `NSAppleEventDescriptor` without creating temporary instances of `NSData`.

The designated initializer for `NSAppleEventDescriptor` is `initWithAEDescNoCopy:` (page 19). However, it is unlikely that you will need to create a subclass of `NSAppleEventDescriptor`.

Cocoa doesn’t currently provide a mechanism for applications to directly send raw Apple events (though compiling and executing an AppleScript script with `NSAppleScript` may result in Apple events being sent). However, Cocoa applications have full access to the Apple Event Manager C APIs for working with Apple events. So, for example, you might use an instance of `NSAppleEventDescriptor` to assemble an Apple event and call the Apple Event Manager function `AESend` to send it.

If you need to send Apple events, or if you need more information on some of the Apple event concepts described here, see *Apple Events Programming Guide* and *Apple Event Manager Reference*.

Adopted Protocols

NSCopying

- `copyWithZone:`

Tasks

Creating and Initializing Descriptors

- + `appleEventWithEventClass:eventID:targetDescriptor:returnID:transactionID:` (page 9)
Creates a descriptor that represents an Apple event, initialized according to the specified information.
- + `descriptorWithBoolean:` (page 10)
Creates a descriptor initialized with type `typeBoolean` that stores the specified Boolean value.
- + `descriptorWithDescriptorType:bytes:length:` (page 10)
Creates a descriptor initialized with the specified event type that stores the specified data (from a series of bytes).

- + [descriptorWithDescriptorType:data:](#) (page 11)
Creates a descriptor initialized with the specified event type that stores the specified data (from an instance of `NSData`).
- + [descriptorWithEnumCode:](#) (page 11)
Creates a descriptor initialized with type `typeEnumerated` that stores the specified enumerator data type value.
- + [descriptorWithInt32:](#) (page 12)
Creates a descriptor initialized with Apple event type `typeSInt32` that stores the specified integer value.
- + [descriptorWithString:](#) (page 12)
Creates a descriptor initialized with type `typeUnicodeText` that stores the text from the specified string.
- + [descriptorWithTypeCode:](#) (page 13)
Creates a descriptor initialized with type `typeType` that stores the specified type value.
- + [listDescriptor](#) (page 13)
Creates and initializes an empty list descriptor.
- + [nullDescriptor](#) (page 13)
Creates and initializes a descriptor with no parameter or attribute values set.
- + [recordDescriptor](#) (page 14)
Creates and initializes a descriptor for an Apple event record whose data has yet to be set.
- [initWithListDescriptor](#) (page 18)
Initializes a newly allocated instance as an empty list descriptor.
- [initWithRecordDescriptor](#) (page 19)
Initializes a newly allocated instance as a descriptor that is an Apple event record.
- [initWithAEDescNoCopy:](#) (page 19)
Initializes a newly allocated instance as a descriptor for the specified Carbon `AEDesc` structure.
- [initWithDescriptorType:bytes:length:](#) (page 20)
Initializes a newly allocated instance as a descriptor with the specified descriptor type and data (from an arbitrary sequence of bytes and a length count).
- [initWithDescriptorType:data:](#) (page 20)
Initializes a newly allocated instance as a descriptor with the specified descriptor type and data (from an instance of `NSData`).
- [initWithEventClass:eventID:targetDescriptor:returnID:transactionID:](#) (page 21)
Initializes a newly allocated instance as a descriptor for an Apple event, initialized with the specified values.

Getting Information About a Descriptor

- [aeDesc](#) (page 14)
Returns a pointer to the `AEDesc` structure that is encapsulated by the receiver, if it has one.
- [booleanValue](#) (page 15)
Returns the contents of the receiver as a Boolean value, coercing (to `typeBoolean`) if necessary.
- [coerceToDescriptorType:](#) (page 15)
Returns a descriptor obtained by coercing the receiver to the specified type.

- [data](#) (page 16)
Returns the receiver's data as an `NSData` object.
- [descriptorType](#) (page 17)
Returns the descriptor type of the receiver.
- [enumCodeValue](#) (page 17)
Returns the contents of the receiver as an enumeration type, coercing (to `typeEnumerated`) if necessary.
- [int32Value](#) (page 22)
Returns the contents of the receiver as an integer, coercing (to `typeSInt32`) if necessary.
- [numberOfItems](#) (page 23)
Returns the number of descriptors in the receiver's descriptor list.
- [stringValue](#) (page 27)
Returns the contents of the receiver as a Unicode text string, coercing (to `typeUnicodeText`) if necessary.
- [typeCodeValue](#) (page 27)
Returns the contents of the receiver as a type, coercing (to `typeType`) if necessary.

Working With List Descriptors

- [descriptorAtIndex:](#) (page 16)
Returns the descriptor at the specified (one-based) position in the receiving descriptor list.
- [insertDescriptorAtIndex:](#) (page 21)
Inserts a descriptor at the specified (one-based) position in the receiving descriptor list, replacing the existing descriptor, if any, at that position.
- [removeDescriptorAtIndex:](#) (page 24)
Removes the descriptor at the specified (one-based) position in the receiving descriptor list.

Working With Record Descriptors

- [descriptorForKeyword:](#) (page 17)
Returns the receiver's descriptor for the specified keyword.
- [keywordForDescriptorAtIndex:](#) (page 22)
Returns the keyword for the descriptor at the specified (one-based) position in the receiver.
- [removeDescriptorWithKeyword:](#) (page 24)
Removes the receiver's descriptor identified by the specified keyword.
- [setDescriptorForKeyword:](#) (page 26)
Adds a descriptor, identified by a keyword, to the receiver.

Working With Apple Event Descriptors

- [attributeDescriptorForKeyword:](#) (page 15)
Returns a descriptor for the receiver's Apple event attribute identified by the specified keyword.

- [eventClass](#) (page 18)
Returns the event class for the receiver.
- [eventID](#) (page 18)
Returns the event ID for the receiver.
- [paramDescriptorForKeyword:](#) (page 23)
Returns a descriptor for the receiver's Apple event parameter identified by the specified keyword.
- [removeParamDescriptorWithKeyword:](#) (page 24)
Removes the receiver's parameter descriptor identified by the specified keyword.
- [returnID](#) (page 25)
Returns the receiver's return ID (the ID for a reply Apple event).
- [setAttributeDescriptor:forKeyword:](#) (page 25)
Adds a descriptor to the receiver as an attribute identified by the specified keyword.
- [setParamDescriptor:forKeyword:](#) (page 26)
Adds a descriptor to the receiver as an Apple event parameter identified by the specified keyword.
- [transactionID](#) (page 27)
Returns the receiver's transaction ID, if any.

Class Methods

appleEventWithEventClass:eventID:targetDescriptor:returnID:transactionID:

Creates a descriptor that represents an Apple event, initialized according to the specified information.

```
+ (NSAppleEventDescriptor *)appleEventWithEventClass:(AEEEventClass)eventClass
  eventID:(AEEEventID)eventID targetDescriptor:(NSAppleEventDescriptor
  *)addressDescriptor returnID:(AEReturnID)returnID
  transactionID:(AETransactionID)transactionID
```

Parameters

eventClass

The event class to be set in the returned descriptor.

eventID

The event ID to be set in the returned descriptor.

addressDescriptor

A pointer to a descriptor that identifies the target application for the Apple event. Passing `nil` results in an Apple event descriptor that has no `keyAddressAttr` attribute (it is valid for an Apple event to have no target address attribute).

returnID

The return ID to be set in the returned descriptor. If you pass a value of `kAutoGenerateReturnID`, the Apple Event Manager assigns the created Apple event a return ID that is unique to the current session. If you pass any other value, the Apple Event Manager assigns that value for the ID.

transactionID

The transaction ID to be set in the returned descriptor. A transaction is a sequence of Apple events that are sent back and forth between client and server applications, beginning with the client's initial request for a service. All Apple events that are part of a transaction must have the same transaction ID. You can specify `kAnyTransactionID` if the Apple event is not one of a series of interdependent Apple events.

Return Value

A descriptor for an Apple event, initialized according to the specified parameter values, or `nil` if an error occurs.

Discussion

Constants such as `kAutoGenerateReturnID` and `kAnyTransactionID` are defined in `AE.framework`, a subframework of `ApplicationServices.framework`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSAppleEventDescriptor.h`

descriptorWithBoolean:

Creates a descriptor initialized with type `typeBoolean` that stores the specified Boolean value.

```
+ (NSAppleEventDescriptor *)descriptorWithBoolean:(Boolean)boolean
```

Parameters

boolean

The Boolean value to be set in the returned descriptor.

Return Value

A descriptor with the specified Boolean value, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSAppleEventDescriptor.h`

descriptorWithDescriptorType:bytes:length:

Creates a descriptor initialized with the specified event type that stores the specified data (from a series of bytes).

```
+ (NSAppleEventDescriptor *)descriptorWithDescriptorType:(DescType)descriptorType
    bytes:(const void *)bytes length:(NSUInteger)byteCount
```

Parameters

descriptorType

The descriptor type to be set in the returned descriptor.

bytes

The data, as a sequence of bytes, to be set in the returned descriptor.

byteCount

The length, in bytes, of the data to be set in the returned descriptor.

Return Value

A descriptor with the specified type and data, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Sketch+Accessibility

Declared In

NSAppleEventDescriptor.h

descriptorWithDescriptorType:data:

Creates a descriptor initialized with the specified event type that stores the specified data (from an instance of `NSData`).

```
+ (NSAppleEventDescriptor *)descriptorWithDescriptorType:(DescType)descriptorType
  data:(NSData *)data
```

Parameters

descriptorType

The descriptor type to be set in the returned descriptor.

data

The data, as an instance of `NSData`, to be set in the returned descriptor.

Return Value

A descriptor with the specified type and data, or `nil` if an error occurs.

Discussion

You can use this method to create a descriptor that you can build into a complete Apple event by calling methods such as [setAttributeDescriptor:forKeyword:](#) (page 25), [setDescriptor:forKeyword:](#) (page 26), and [setParamDescriptor:forKeyword:](#) (page 26).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

SimpleScriptingPlugin

Declared In

NSAppleEventDescriptor.h

descriptorWithEnumCode:

Creates a descriptor initialized with type `typeEnumerated` that stores the specified enumerator data type value.

```
+ (NSAppleEventDescriptor *)descriptorWithEnumCode:(OSType)enumerator
```

Parameters*enumerator*

A type code that identifies the type of enumerated data to be stored in the returned descriptor.

Return Value

A descriptor with the specified enumerator data type value, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSAppleEventDescriptor.h

descriptorWithInt32:

Creates a descriptor initialized with Apple event type `typeSInt32` that stores the specified integer value.

```
+ (NSAppleEventDescriptor *)descriptorWithInt32:(SInt32)signedInt
```

Parameters*signedInt*

The integer value to be stored in the returned descriptor.

Return Value

A descriptor containing the specified integer value, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

AttachAScript

Sketch-112

Declared In

NSAppleEventDescriptor.h

descriptorWithString:

Creates a descriptor initialized with type `typeUnicodeText` that stores the text from the specified string.

```
+ (NSAppleEventDescriptor *)descriptorWithString:(NSString *)string
```

Parameters*string*

A string that specifies the text to be stored in the returned descriptor.

Return Value

A descriptor that contains the text from the specified string, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

AttachAScript

Declared In

NSAppleEventDescriptor.h

descriptorWithTypeCode:Creates a descriptor initialized with type `typeType` that stores the specified type value.+ (NSAppleEventDescriptor *)descriptorWithTypeCode:(OSType) *typeCode***Parameters***typeCode*

The type value to be set in the returned descriptor.

Return ValueA descriptor with the specified type, or `nil` if an error occurs.**Availability**

Available in Mac OS X v10.2 and later.

Declared In

NSAppleEventDescriptor.h

listDescriptor

Creates and initializes an empty list descriptor.

+ (NSAppleEventDescriptor *)listDescriptor

Return ValueAn empty list descriptor, or `nil` if an error occurs.**Discussion**

A list descriptor is a descriptor whose data consists of one or more descriptors. You can add items to the list by calling [insertDescriptorAtIndex:](#) (page 21) or remove them with [removeDescriptorAtIndex:](#) (page 24).

Invoking this method is equivalent to allocating an instance of `NSAppleEventDescriptor` and invoking [initListDescriptor](#) (page 18).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AttachAScript

Declared In

NSAppleEventDescriptor.h

nullDescriptor

Creates and initializes a descriptor with no parameter or attribute values set.

+ (NSAppleEventDescriptor *)nullDescriptor

Return Value

A descriptor with no parameter or attribute values set, or `nil` if an error occurs.

Discussion

You don't typically call this method, as most `NSAppleEventDescriptor` instance methods can't be safely called on the returned empty descriptor.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSAppleEventDescriptor.h`

recordDescriptor

Creates and initializes a descriptor for an Apple event record whose data has yet to be set.

```
+ (NSAppleEventDescriptor *)recordDescriptor
```

Return Value

An Apple event descriptor whose data has yet to be set, or `nil` if an error occurs.

Discussion

An Apple event record is a descriptor whose data is a set of descriptors keyed by four-character codes. You can add information to the descriptor with methods such as [setAttributeDescriptor:forKeyword:](#) (page 25), [setDescription:forKeyword:](#) (page 26), and [setParameterDescriptor:forKeyword:](#) (page 26).

Invoking this method is equivalent to allocating an instance of `NSAppleEventDescriptor` and invoking [initWithRecordDescriptor:](#) (page 19).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSAppleEventDescriptor.h`

Instance Methods

aeDesc

Returns a pointer to the `AEDesc` structure that is encapsulated by the receiver, if it has one.

```
- (const AEDesc *)aeDesc
```

Return Value

If the receiver has a valid `AEDesc` structure, returns a pointer to it; otherwise returns `nil`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSAppleEventDescriptor.h`

attributeDescriptorForKeyword:

Returns a descriptor for the receiver's Apple event attribute identified by the specified keyword.

- (NSAppleEventDescriptor *)attributeDescriptorForKeyword:(AEKeyword)keyword

Parameters

keyword

A keyword (a four-character code) that identifies the descriptor to obtain.

Return Value

The attribute descriptor for the specified keyword, or `nil` if an error occurs.

Discussion

The receiver must be an Apple event.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

booleanValue

Returns the contents of the receiver as a Boolean value, coercing (to `typeBoolean`) if necessary.

- (Boolean)booleanValue

Return Value

The contents of the descriptor, as a Boolean value, or `false` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Apply Firmware Password

Declared In

NSAppleEventDescriptor.h

coerceToDescriptorType:

Returns a descriptor obtained by coercing the receiver to the specified type.

- (NSAppleEventDescriptor *)coerceToDescriptorType:(DescType)descriptorType

Parameters

descriptorType

The descriptor type to coerce the receiver to.

Return Value

A descriptor of the specified type, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Sketch+Accessibility

Declared In

NSAppleEventDescriptor.h

dataReturns the receiver's data as an `NSData` object.

- (NSData *)data

Return ValueAn instance of `NSData` containing the receiver's data, or `nil` if an error occurs.**Availability**

Available in Mac OS X v10.0 and later.

Related Sample Code

Apply Firmware Password

Sketch+Accessibility

Declared In

NSAppleEventDescriptor.h

descriptorAtIndex:

Returns the descriptor at the specified (one-based) position in the receiving descriptor list.

- (NSAppleEventDescriptor *)descriptorAtIndex:(NSInteger)anIndex

Parameters*anIndex*

The one-based descriptor list position of the descriptor to return.

Return ValueThe descriptor from the specified position (one-based) in the descriptor list, or `nil` if the specified descriptor cannot be obtained.**Availability**

Available in Mac OS X v10.0 and later.

See Also- [insertDescriptorAtIndex:](#) (page 21)- [removeDescriptorAtIndex:](#) (page 24)**Related Sample Code**

Apply Firmware Password

AttachAScript

Declared In

NSAppleEventDescriptor.h

descriptorForKeyword:

Returns the receiver's descriptor for the specified keyword.

- (NSAppleEventDescriptor *)descriptorForKeyword:(AEKeyword) keyword

Parameters

keyword

A keyword (a four-character code) that identifies the descriptor to obtain.

Return Value

A descriptor for the specified keyword, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

descriptorType

Returns the descriptor type of the receiver.

- (DescType)descriptorType

Return Value

The descriptor type of the receiver.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

enumCodeValue

Returns the contents of the receiver as an enumeration type, coercing (to `typeEnumerated`) if necessary.

- (OSType)enumCodeValue

Return Value

The contents of the descriptor, as an enumeration type, or 0 if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Apply Firmware Password

Declared In

NSAppleEventDescriptor.h

eventClass

Returns the event class for the receiver.

```
- (AEventClass)eventClass
```

Return Value

The event class (a four-character code) for the receiver, or 0 if an error occurs.

Discussion

The receiver must be an Apple event. An Apple event is identified by its event class and event ID, a pair of four-character codes stored as 32-bit integers. For example, most events in the Standard suite have the four-character code 'core' (defined as the constant `kAECoreSuite` in `AE.framework`, a subframework of `ApplicationServices.framework`). For more information on event classes and event IDs, see *Building an Apple Event* in *Apple Events Programming Guide*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSAppleEventDescriptor.h`

eventID

Returns the event ID for the receiver.

```
- (AEventID)eventID
```

Return Value

The event ID (a four-character code) for the receiver, or 0 if an error occurs.

Discussion

The receiver must be an Apple event. An Apple event is identified by its event class and event ID, a pair of four-character codes stored as 32-bit integers. For example, the open Apple event from the Standard suite has the four-character code 'odoc' (defined as the constant `kAEOpen` in `AE.framework`, a subframework of `ApplicationServices.framework`).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSAppleEventDescriptor.h`

initListDescriptor

Initializes a newly allocated instance as an empty list descriptor.

```
- (id)initListDescriptor
```

Return Value

An empty list descriptor, or `nil` if an error occurs.

Discussion

You can add items to the empty list descriptor with [insertDescriptorAtIndex:](#) (page 21). The list indices are one-based.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [listDescriptor](#) (page 13)

Declared In

NSAppleEventDescriptor.h

initRecordDescriptor

Initializes a newly allocated instance as a descriptor that is an Apple event record.

```
- (id)initRecordDescriptor
```

Return Value

The initialized Apple event record, or `nil` if an error occurs.

Discussion

An Apple event record is a descriptor whose data is a set of descriptors keyed by four-character codes. You can add information to the descriptor with methods such as [setAttributeDescriptor:forKeyword:](#) (page 25), [setDescriptor:forKeyword:](#) (page 26), and [setParamDescriptor:forKeyword:](#) (page 26).

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [recordDescriptor](#) (page 14)

Declared In

NSAppleEventDescriptor.h

initWithAEDescNoCopy:

Initializes a newly allocated instance as a descriptor for the specified Carbon `AEDesc` structure.

```
- (id)initWithAEDescNoCopy:(const AEDesc *)aeDesc
```

Parameters

aeDesc

A pointer to the `AEDesc` structure to associate with the descriptor.

Return Value

An instance of `NSAppleEventDescriptor` that is associated with the structure pointed to by *aeDesc*, or `nil` if an error occurs.

Discussion

The initialized object takes responsibility for calling the `AEDisposeDesc` function on the `AEDesc` at object deallocation time. This is the designated initializer for this class.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSAppleEventDescriptor.h

initWithDescriptorType:bytes:length:

Initializes a newly allocated instance as a descriptor with the specified descriptor type and data (from an arbitrary sequence of bytes and a length count).

```
- (id)initWithDescriptorType:(DescType)descriptorType bytes:(const void *)bytes
    length:(NSUInteger)byteCount
```

Parameters

descriptorType

The descriptor type to be set in the returned descriptor.

bytes

The data, as a sequence of bytes, to be set in the returned descriptor.

byteCount

The length, in bytes, of the data to be set in the returned descriptor.

Return Value

An instance of `NSAppleEventDescriptor` with the specified type and data. Returns `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSAppleEventDescriptor.h

initWithDescriptorType:data:

Initializes a newly allocated instance as a descriptor with the specified descriptor type and data (from an instance of `NSData`).

```
- (id)initWithDescriptorType:(DescType)descriptorType data:(NSData *)data
```

Parameters

descriptorType

The descriptor type to be set in the initialized descriptor.

data

The data to be set in the initialized descriptor.

Return Value

An instance of `NSAppleEventDescriptor` with the specified type and data. Returns `nil` if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [descriptorWithDescriptorType:data:](#) (page 11)

Declared In

NSAppleEventDescriptor.h

initWithEventClass:eventID:targetDescriptor:returnID:transactionID:

Initializes a newly allocated instance as a descriptor for an Apple event, initialized with the specified values.

```
- (id)initWithEventClass:(AEEEventClass)eventClass eventID:(AEEEventID)eventID
  targetDescriptor:(NSAppleEventDescriptor *)addressDescriptor
  returnID:(AEReturnID)returnID transactionID:(AETransactionID)transactionID
```

Parameters*eventClass*

The event class to be set in the returned descriptor.

eventID

The event ID to be set in the returned descriptor.

addressDescriptor

A pointer to a descriptor that identifies the target application for the Apple event. Passing `nil` results in an Apple event descriptor that has no `keyAddressAttr` attribute (it is valid for an Apple event to have no target address attribute).

returnID

The return ID to be set in the returned descriptor. If you pass a value of `kAutoGenerateReturnID`, the Apple Event Manager assigns the created Apple event a return ID that is unique to the current session. If you pass any other value, the Apple Event Manager assigns that value for the ID.

transactionID

The transaction ID to be set in the returned descriptor. A transaction is a sequence of Apple events that are sent back and forth between client and server applications, beginning with the client's initial request for a service. All Apple events that are part of a transaction must have the same transaction ID. You can specify `kAnyTransactionID` if the Apple event is not one of a series of interdependent Apple events.

Return Value

The initialized Apple event (an instance of `NSAppleEventDescriptor`), or `nil` if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

insertDescriptor:atIndex:

Inserts a descriptor at the specified (one-based) position in the receiving descriptor list, replacing the existing descriptor, if any, at that position.

```
- (void)insertDescriptor:(NSAppleEventDescriptor *)descriptor
  atIndex:(NSInteger)anIndex
```

Parameters*descriptor*

The descriptor to insert in the receiver. Specifying an index of 0 or count + 1 causes appending to the end of the list.

anIndex

The one-based descriptor list position at which to insert the descriptor.

Discussion

Because it actually replaces the descriptor, if any, at the specified position, this method might better be called `replaceDescriptorAtIndex:`. The receiver must be a list descriptor. The indices are one-based. Currently provides no indication if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [descriptorAtIndex:](#) (page 16)
- [removeDescriptorAtIndex:](#) (page 24)

Related Sample Code

AttachAScript

Declared In

NSAppleEventDescriptor.h

int32Value

Returns the contents of the receiver as an integer, coercing (to `typeSInt32`) if necessary.

- (`SInt32`)int32Value

Return Value

The contents of the descriptor, as an integer value, or 0 if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Apply Firmware Password
AttachAScript

Declared In

NSAppleEventDescriptor.h

keywordForDescriptorAtIndex:

Returns the keyword for the descriptor at the specified (one-based) position in the receiver.

- (`AEKeyword`)keywordForDescriptorAtIndex:(`NSInteger`)*anIndex*

Parameters*anIndex*

The one-based descriptor list position of the descriptor to get the keyword for.

Return Value

The keyword (a four-character code) for the descriptor at the one-based location specified by *anIndex*, or 0 if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

numberOfItems

Returns the number of descriptors in the receiver's descriptor list.

- (NSInteger)numberOfItems

Return Value

The number of descriptors in the receiver's descriptor list (possibly 0); returns 0 if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Apply Firmware Password

Declared In

NSAppleEventDescriptor.h

paramDescriptorForKeyword:

Returns a descriptor for the receiver's Apple event parameter identified by the specified keyword.

- (NSAppleEventDescriptor *)paramDescriptorForKeyword:(AEKeyword)keyword

Parameters*keyword*

A keyword (a four-character code) that identifies the parameter descriptor to obtain.

Return Value

A descriptor for the specified keyword, or `nil` if an error occurs.

Discussion

The receiver must be an Apple event.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CoreRecipes

Declared In

NSAppleEventDescriptor.h

removeDescriptorAtIndex:

Removes the descriptor at the specified (one-based) position in the receiving descriptor list.

```
- (void)removeDescriptorAtIndex:(NSInteger)anIndex
```

Parameters*anIndex*

The one-based position of the descriptor to remove.

Discussion

The receiver must be a list descriptor. The indices are one-based. Currently provides no indication if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [descriptorAtIndex:](#) (page 16)
- [insertDescriptorAtIndex:](#) (page 21)

Declared In

NSAppleEventDescriptor.h

removeDescriptorWithKeyword:

Removes the receiver's descriptor identified by the specified keyword.

```
- (void)removeDescriptorWithKeyword:(AEKeyword)keyword
```

Parameters*keyword*

A keyword (a four-character code) that identifies the descriptor to remove.

Discussion

The receiver must be an Apple event or Apple event record. Currently provides no indication if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

removeParamDescriptorWithKeyword:

Removes the receiver's parameter descriptor identified by the specified keyword.

```
- (void)removeParamDescriptorWithKeyword:(AEKeyword)keyword
```

Parameters*keyword*

A keyword (a four-character code) that identifies the parameter descriptor to remove. Currently provides no indication if an error occurs.

Discussion

The receiver must be an Apple event or Apple event record, both of which can contain parameters.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

returnID

Returns the receiver's return ID (the ID for a reply Apple event).

```
- (AEReturnID) returnID
```

Return Value

The receiver's return ID (an integer value), or 0 if an error occurs.

Discussion

The receiver must be an Apple event.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

setAttributeDescriptor:forKeyword:

Adds a descriptor to the receiver as an attribute identified by the specified keyword.

```
- (void) setAttributeDescriptor:(NSAppleEventDescriptor *) descriptor
    forKeyword:(AEKeyword) keyword
```

Parameters*descriptor*

The attribute descriptor to add to the receiver.

keyword

A keyword (a four-character code) that identifies the attribute descriptor to add. If a descriptor with that keyword already exists in the receiver, it is replaced.

Discussion

The receiver must be an Apple event. Currently provides no indication if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

setDescription:forKeyword:

Adds a descriptor, identified by a keyword, to the receiver.

```
- (void)setDescriptor:(NSAppleEventDescriptor *)descriptor
    forKeyword:(AEKeyword)keyword
```

Parameters

descriptor

The descriptor to add to the receiver.

keyword

A keyword (a four-character code) that identifies the descriptor to add. If a descriptor with that keyword already exists in the receiver, it is replaced.

Discussion

The receiver must be an Apple event or Apple event record. Currently provides no indication if an error occurs.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AttachAScript

SimpleScriptingPlugin

Sketch-112

Declared In

NSAppleEventDescriptor.h

setParameterDescriptor:forKeyword:

Adds a descriptor to the receiver as an Apple event parameter identified by the specified keyword.

```
- (void)setParamDescriptor:(NSAppleEventDescriptor *)descriptor
    forKeyword:(AEKeyword)keyword
```

Parameters

descriptor

The parameter descriptor to add to the receiver.

keyword

A keyword (a four-character code) that identifies the parameter descriptor to add. If a descriptor with that keyword already exists in the receiver, it is replaced.

Discussion

The receiver must be an Apple event or Apple event record, both of which can contain parameters.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

stringValue

Returns the contents of the receiver as a Unicode text string, coercing (to `typeUnicodeText`) if necessary.

```
- (NSString *)stringValue
```

Return Value

The contents of the descriptor, as a string, or `nil` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Apply Firmware Password

AttachAScript

CoreRecipes

Declared In

NSAppleEventDescriptor.h

transactionID

Returns the receiver's transaction ID, if any.

```
- (AETransactionID)transactionID
```

Return Value

The receiver's transaction ID (an integer value), or 0 if an error occurs.

Discussion

The receiver must be an Apple event. Currently provides no indication if an error occurs. For more information on transactions, see the description for [appleEventWithEventClass:eventID:targetDescriptor:returnID:transactionID:](#) (page 9).

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSAppleEventDescriptor.h

typeCodeValue

Returns the contents of the receiver as a type, coercing (to `typeType`) if necessary.

```
- (OSType)typeCodeValue
```

Return Value

The contents of the descriptor, as a type, or 0 if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

Apply Firmware Password

Declared In

NSAppleEventDescriptor.h

Document Revision History

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

Date	Notes
2007-04-10	Updated parameter declarations to reflect use of NSInteger and NSUInteger types.
2006-11-07	Added information to Class Description and revised parameter descriptions.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

aeDesc **instance method** 14
appleEventWithEventClass:eventID:targetDescriptor:
returnID:transactionID: **class method** 9
attributeDescriptorForKeyword: **instance method**
15

B

booleanValue **instance method** 15

C

coerceToDescriptorType: **instance method** 15

D

data **instance method** 16
descriptorAtIndex: **instance method** 16
descriptorForKeyword: **instance method** 17
descriptorType **instance method** 17
descriptorWithBoolean: **class method** 10
descriptorWithDescriptorType:bytes:length:
class method 10
descriptorWithDescriptorType:data: **class method**
11
descriptorWithEnumCode: **class method** 11
descriptorWithInt32: **class method** 12
descriptorWithString: **class method** 12
descriptorWithTypeCode: **class method** 13

E

enumCodeValue **instance method** 17

eventClass **instance method** 18

eventID **instance method** 18

I

initListDescriptor **instance method** 18
initRecordDescriptor **instance method** 19
initWithAEDescNoCopy: **instance method** 19
initWithDescriptorType:bytes:length: **instance**
method 20
initWithDescriptorType:data: **instance method** 20
initWithEventClass:eventID:targetDescriptor:
returnID:transactionID: **instance method** 21
insertDescriptor:atIndex: **instance method** 21
int32Value **instance method** 22

K

keywordForDescriptorAtIndex: **instance method** 22

L

listDescriptor **class method** 13

N

nullDescriptor **class method** 13
numberOfItems **instance method** 23

P

paramDescriptorForKeyword: **instance method** 23

R

recordDescriptor **class method** [14](#)
removeDescriptorAtIndex: **instance method** [24](#)
removeDescriptorWithKeyword: **instance method** [24](#)
removeParamDescriptorWithKeyword: **instance method** [24](#)
returnID **instance method** [25](#)

S

setAttributeDescriptor:forKeyword: **instance method** [25](#)
setDescription:forKeyword: **instance method** [26](#)
setParamDescriptor:forKeyword: **instance method** [26](#)
stringValue **instance method** [27](#)

T

transactionID **instance method** [27](#)
typeCodeValue **instance method** [27](#)