NSAppleEventDescriptor Class Reference
Contents

**NSAppleEventDescriptor Class Reference**  4
Overview 4
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: 11
  descriptorWithDescriptorType:data: 11
  descriptorWithEnumCode: 12
  descriptorWithInt32: 13
  descriptorWithString: 13
  descriptorWithTypeCode: 14
  listDescriptor 14
  nullDescriptor 15
  recordDescriptor 15
Instance Methods 16
  aeDesc 16
  attributeDescriptorForKeyword: 16
  booleanValue 16
  coerceToDescriptorType: 17
  data 17
  descriptorAtIndex: 18
  descriptorForKeyword: 19
  descriptorType 19
  enumCodeValue 19
  eventClass 20
  eventID 20
  initListDescriptor 21
# NSAppleEventDescriptor Class Reference

Inherits from | NSObject
--- | ---
Conforms to | NSCopying
 | NSSecureCoding
 | NSObject (NSObject)
Framework | /System/Library/Frameworks/Foundation.framework
Availability | Available in OS X v10.0 and later.
Companion guide | Cocoa Scripting Guide
Declared in | NSAppleEventDescriptor.h
Related sample code | Apply Firmware Password
 | AttachAScript
 | SimpleScriptingPlugin
 | Sketch
 | 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 16) method. Other methods, including `descriptorWithDescriptorType:bytes:length:` (page 11) make it possible to create and initialize instances of NSAppleEventDescriptor without creating temporary instances of NSData.

The designated initializer for NSAppleEventDescriptor is `initWithAEDescNoCopy:` (page 22). 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 11)
   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 12)
   Creates a descriptor initialized with type typeEnumerated that stores the specified enumerator data type value.

+ descriptorWithInt32: (page 13)
   Creates a descriptor initialized with Apple event type typeSInt32 that stores the specified integer value.

+ descriptorWithString: (page 13)
   Creates a descriptor initialized with type typeUnicodeText that stores the text from the specified string.

+ descriptorWithTypeCode: (page 14)
   Creates a descriptor initialized with type typeType that stores the specified type value.

+ listDescriptor (page 14)
   Creates and initializes an empty list descriptor.

+ nullDescriptor (page 15)
   Creates and initializes a descriptor with no parameter or attribute values set.

+ recordDescriptor (page 15)
   Creates and initializes a descriptor for an Apple event record whose data has yet to be set.
- **initListDescriptor** (page 21)
  Initializes a newly allocated instance as an empty list descriptor.

- **initRecordDescriptor** (page 21)
  Initializes a newly allocated instance as a descriptor that is an Apple event record.

- **initWithAEDescNoCopy:** (page 22)
  Initializes a newly allocated instance as a descriptor for the specified Carbon AEDesc structure.

- **initWithDescriptorType:bytes:length:** (page 23)
  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 23)
  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 24)
  Initializes a newly allocated instance as a descriptor for an Apple event, initialized with the specified values.

**Getting Information About a Descriptor**

- **aeDesc** (page 16)
  Returns a pointer to the AEDesc structure that is encapsulated by the receiver, if it has one.

- **booleanValue** (page 16)
  Returns the contents of the receiver as a Boolean value, coercing (to typeBoolean) if necessary.

- **coerceToDescriptorType:** (page 17)
  Returns a descriptor obtained by coercing the receiver to the specified type.

- **data** (page 17)
  Returns the receiver’s data as an NSData object.

- **descriptorType** (page 19)
  Returns the descriptor type of the receiver.

- **enumCodeValue** (page 19)
  Returns the contents of the receiver as an enumeration type, coercing (to typeEnumerated) if necessary.

- **int32Value** (page 25)
  Returns the contents of the receiver as an integer, coercing (to typeSInt32) if necessary.

- **numberOfItems** (page 26)
  Returns the number of descriptors in the receiver’s descriptor list.
– **stringValue** (page 31)
  Returns the contents of the receiver as a Unicode text string, coercing (to typeUnicodeText) if necessary.

– **typeCodeValue** (page 32)
  Returns the contents of the receiver as a type, coercing (to typeType) if necessary.

### Working With List Descriptors

– **descriptorAtIndex**: (page 18)
  Returns the descriptor at the specified (one-based) position in the receiving descriptor list.

– **insertDescriptor:atIndex**: (page 25)
  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 27)
  Removes the descriptor at the specified (one-based) position in the receiving descriptor list.

### Working With Record Descriptors

– **descriptorForKeyword**: (page 19)
  Returns the receiver’s descriptor for the specified keyword.

– **keywordForDescriptorAtIndex**: (page 26)
  Returns the keyword for the descriptor at the specified (one-based) position in the receiver.

– **removeDescriptorWithKeyword**: (page 28)
  Removes the receiver’s descriptor identified by the specified keyword.

– **setDescriptor:forKeyword**: (page 30)
  Adds a descriptor, identified by a keyword, to the receiver.

### Working With Apple Event Descriptors

– **attributeDescriptorForKeyword**: (page 16)
  Returns a descriptor for the receiver’s Apple event attribute identified by the specified keyword.

– **eventClass** (page 20)
  Returns the event class for the receiver.
– **eventID** (page 20)
  Returns the event ID for the receiver.

– **paramDescriptorForKeyword:** (page 27)
  Returns a descriptor for the receiver’s Apple event parameter identified by the specified keyword.

– **removeParamDescriptorWithKeyword:** (page 28)
  Removes the receiver’s parameter descriptor identified by the specified keyword.

– **returnID** (page 29)
  Returns the receiver’s return ID (the ID for a reply Apple event).

– **setAttributeDescriptor:forKeyword:** (page 29)
  Adds a descriptor to the receiver as an attribute identified by the specified keyword.

– **setParamDescriptor:forKeyword:** (page 31)
  Adds a descriptor to the receiver as an Apple event parameter identified by the specified keyword.

– **transactionID** (page 32)
  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.*

```objc
+(NSAppleEventDescriptor *)appleEventWithEventClass:(AEEventClass)eventClass
eventID:(AEEventID)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 OS X v10.0 and later.

Related Sample Code
AttachAScript

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 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 OS X v10.2 and later.

Related Sample Code
AttachAScript
Sketch
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 29), `setDescriptor:forKeyword:` (page 30), and `setParamDescriptor:forKeyword:` (page 31).

Availability

Available in 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 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.

```objective-c
+ (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 OS X v10.2 and later.

**Related Sample Code**

AttachAScript

**Declared in**

`NSAppleEventDescriptor.h`

---

**descriptorWithString:**

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

```objective-c
+ (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 OS X v10.2 and later.

**Related Sample Code**

AttachAScript

**Declared in**

`NSAppleEventDescriptor.h`
**descriptorWithWithTypeCode:**

_Creates a descriptor initialized with type _typeType that stores the specified type value._

+ (NSAppleEventDescriptor *)descriptorWithWithTypeCode:(OSType)typeCode

**Parameters**

_typeCode_

The type value to be set in the returned descriptor.

**Return Value**

A descriptor with the specified type, or _nil_ if an error occurs.

**Availability**

Available in OS X v10.2 and later.

**Declared in**

NSAppleEventDescriptor.h

**listDescriptor**

_Creates and initializes an empty list descriptor._

+ (NSAppleEventDescriptor *)listDescriptor

**Return Value**

An 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 25) or remove them with _removeDescriptorAtIndex:_ (page 27).

Invoking this method is equivalent to allocating an instance of _NSAppleEventDescriptor_ and invoking _initWithListDescriptor_ (page 21).

**Availability**

Available in 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 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 29), setDescriptor:forKeyword: (page 30), and setParamDescriptor:forKeyword: (page 31).

Invoking this method is equivalent to allocating an instance of NSAppleEventDescriptor and invoking initRecordDescriptor (page 21).

Availability
Available in 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 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 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.*
Return Value
The contents of the descriptor, as a Boolean value, or `false` if an error occurs.

Availability
Available in OS X v10.2 and later.

Related Sample Code
Apply Firmware Password

Declared in
NSAppleEventDescriptor.h

correctToDescriptorType:

_Returns a descriptor obtained by coercing the receiver to the specified type._

– (NSAppleEventDescriptor *)correctToDescriptorType:(DescType)descriptorType

Parameters
descrriptorType
  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 OS X v10.0 and later.

Related Sample Code
Sketch
Sketch+Accessibility

Declared in
NSAppleEventDescriptor.h

data

_Returns the receiver’s data as an NSData object._

– (NSData *)data
Return Value
An instance of `NSData` containing the receiver’s data, or `nil` if an error occurs.

Availability
Available in OS X v10.0 and later.

Related Sample Code
Apply Firmware Password
Sketch
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 Value
The descriptor from the specified position (one-based) in the descriptor list, or `nil` if the specified descriptor cannot be obtained.

Availability
Available in OS X v10.0 and later.

See Also
- `insertDescriptor:atIndex:` (page 25)
- `removeDescriptorAtIndex:` (page 27)

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 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 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 OS X v10.2 and later.

Related Sample Code
Apply Firmware Password

Declared in
NSAppleEventDescriptor.h

eventClass

*Returns the event class for the receiver.*

- (AEEventClass)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 OS X v10.0 and later.

Related Sample Code
TextEdit

Declared in
NSAppleEventDescriptor.h

eventID

*Returns the event ID for the receiver.*

- (AEEventID)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 OS X v10.0 and later.

Related Sample Code
TextEdit

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 insertDescriptor:atIndex: (page 25). The list indices are one-based.

Availability
Available in OS X v10.0 and later.

See Also
+ listDescriptor (page 14)

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 29), `setDescriptor:forKeyword:` (page 30), and `setParamDescriptor:forKeyword:` (page 31).

Availability
Available in OS X v10.0 and later.

See Also
+ `recordDescriptor` (page 15)

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 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 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 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:(AEEventClass)eventClass eventID:(AEEventID)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 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 replaceDescriptor:atIndex:. The receiver must be a list descriptor. The indices are one-based. Currently provides no indication if an error occurs.

Availability
Available in OS X v10.0 and later.

See Also
- descriptorAtIndex: (page 18)
- removeDescriptorAtIndex: (page 27)

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 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 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 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 OS X v10.0 and later.

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 OS X v10.2 and later.

See Also
– descriptorAtIndex: (page 18)
– insertDescriptor:atIndex: (page 25)

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 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 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 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 OS X v10.0 and later.

Declared in
NSAppleEventDescriptor.h

setDescriptor: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 OS X v10.0 and later.

Related Sample Code
AttachAScript
SimpleScriptingPlugin

Declared in
NSAppleEventDescriptor.h
**setParamDescriptor: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 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 OS X v10.2 and later.

**Related Sample Code**

Apply Firmware Password
AttachAScript

**Declared in**

NSAppleEventDescriptor.h
**transactionID**

Returns the receiver's transaction ID, if any.

```c
- (NSString *)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 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.

```c
- (OSType)typeCodeValue
```

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

**Availability**
Available in 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*.

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