API Collection

Apple Event Manager

Overview

The Apple Event Manager, a part of the Open Scripting Architecture (OSA), provides facilities for applications to send and respond to Apple events and to make their operations and data available to AppleScript scripts. For related API reference, see Open Scripting Architecture Reference.

An Apple event is a type of interprocess message that can specify complex operations and data. Apple events provide a data transport and event dispatching mechanism that can be used within a single application, between applications on the same computer, and between applications on different computers connected to a network.

Applications typically use Apple events to request services and information from other applications or to provide services and information in response to such requests. All applications that present a graphical interface to the user through the Human Interface Toolbox (Carbon applications) or the Cocoa application framework should be able to respond, if appropriate, to certain events sent by the Mac OS. These include the `open application` (or `launch`), `reopen`, `open documents`, `print documents`, and `quit` events.

Some Apple Event Manager functions are marked as being thread safe—for all other functions, you should call them only on the main thread.

For an overview of technologies that take advantage of the Apple Event Manager, see AppleScript Overview.

For information on working with Apple events, including events sent by the Mac OS, see Responding to Apple Events in Apple Events Programming Guide. For information about individual four-character codes used in Apple events, see AppleScript Terminology and Apple Event Codes Reference.

The Apple Event Manager is implemented by the AE framework, a subframework of the Core Services framework. You don't link directly with the AE framework—instead, you typically link with the Carbon framework, which includes it. Some AppleEvent definitions are only available to clients

of the Carbon framework, which includes, for example, `AEInteraction.h` in the HIToolbox framework.

The AE framework does not force a connection to the window server. This allows daemons and startup items that work with Apple events to continue working across log outs.

Gestalt Constants

You can check for version and feature availabilityinformation by using the Apple Event Manager selectors defined inthe Gestalt Manager. For more information see *Inside macOS: Gestalt Manager Reference*.

Topics

Adding Items to Descriptor Lists

`func AEPutArray(UnsafeMutablePointer<AEDescList>!, AEArrayType, UnsafePointer<AEArrayData>!, DescType, Size, Int) -> OSErr`

Inserts the data for an Apple event array into a descriptor list, replacing any previous descriptors in the list.

`func AEPutDesc(UnsafeMutablePointer<AEDescList>!, Int, UnsafePointer<AEDesc>!) -> OSErr`

Adds a descriptor to any descriptor list, possibly replacing an existing descriptor in the list.

`func AEPutPtr(UnsafeMutablePointer<AEDescList>!, Int, DescType, UnsafeRawPointer!, Size) -> OSErr`

Inserts data specified in a buffer into a descriptor list as a descriptor, possibly replacing an existing descriptor in the list.

Adding Parameters and Attributes to Apple Events and Apple Event Records

`func AEPutAttributeDesc(UnsafeMutablePointer<AppleEvent>!, AEKeyword, UnsafePointer<AEDesc>!) -> OSErr`

Adds a descriptor and a keyword to an Apple event as an attribute.

`func AEPutAttributePtr(UnsafeMutablePointer<AppleEvent>!, AEKeyword, DescType, UnsafeRawPointer!, Size) -> OSErr`

Adds a pointer to data, a descriptor type, and a keyword to an Apple event as an attribute.

```
func AEPutParamDesc(UnsafeMutablePointer<AppleEvent>!, AEKeyword,
UnsafePointer<AEDesc>!) -> OSErr
```

Inserts a descriptor and a keyword into an Apple event or Apple event record as an Apple event parameter.

```
func AEPutParamPtr(UnsafeMutablePointer<AppleEvent>!, AEKeyword, Desc
Type, UnsafeRawPointer!, Size) -> OSErr
```

Inserts data, a descriptor type, and a keyword into an Apple event or Apple event record as an Apple event parameter.

Coercing Descriptor Types

```
func AECoerceDesc(UnsafePointer<AEDesc>!, DescType, UnsafeMutable
Pointer<AEDesc>!) -> OSErr
```

Coerces the data in a descriptor to another descriptor type and creates a descriptor containing the newly coerced data.

```
func AECoercePtr(DescType, UnsafeRawPointer!, Size, DescType, Unsafe
MutablePointer<AEDesc>!) -> OSErr
```

Coerces data to a desired descriptor type and creates a descriptor containing the newly coerced data.

Counting the Items in Descriptor Lists

```
func AECountItems(UnsafePointer<AEDescList>!, UnsafeMutable
Pointer<Int>!) -> OSErr
```

Counts the number of descriptors in a descriptor list.

Creating an Apple Event

```
func AECreateAppleEvent(AEEventClass, AEEventID, Unsafe
Pointer<AEAddressDesc>!, AEReturnID, AETransactionID, UnsafeMutable
Pointer<AppleEvent>!) -> OSErr
```

Creates an Apple event with several important attributes but no parameters.

Creating and Duplicating Descriptors

```
func AECreateDesc(DescType, UnsafeRawPointer!, Size, UnsafeMutable
Pointer<AEDesc>!) -> OSErr
```

Creates a new descriptor that incorporates the specified data.

```
func AECreateDescFromExternalPtr(OSType, UnsafeRawPointer!, Size,
AEDisposeExternalUPP!, SRefCon!, UnsafeMutablePointer<AEDesc>!) ->
OSStatus
```

Creates a new descriptor that uses a memory buffer supplied by the caller.

```
func AEDuplicateDesc(UnsafePointer<AEDesc>!, UnsafeMutable
Pointer<AEDesc>!) -> OSErr
```

Creates a copy of a descriptor.

Creating, Calling, and Deleting Universal Procedure Pointers

```
func DisposeAECoerceDescUPP(AECoerceDescUPP!)
```

Disposes of a universal procedure pointer to a function that coerces data stored in a descriptor.

```
func DisposeAECoercePtrUPP(AECoercePtrUPP!)
```

Disposes of a universal procedure pointer to a function that coerces data stored in a buffer.

```
func DisposeAEDisposeExternalUPP(AEDisposeExternalUPP!)
```

Disposes of a universal procedure pointer to a function that disposes of data supplied to the `AECreateDescFromExternalPtr` function.

```
func DisposeAEEventHandlerUPP(AEEventHandlerUPP!)
```

Disposes of a universal procedure pointer to an event handler function.

```
func DisposeOSLAccessorUPP(OSLAccessorUPP!)
```

Disposes of a universal procedure pointer to an object accessor function.

```
func DisposeOSLAdjustMarksUPP(OSLAdjustMarksUPP!)
```

Disposes of a universal procedure pointer to an object callback adjust marks function.

```
func DisposeOSLCompareUPP(OSLCompareUPP!)
```

Disposes of a universal procedure pointer to an object callback comparison function.

```
func DisposeOSLCountUPP(OSLCountUPP!)
```

Disposes of a universal procedure pointer to an object callback count function.

```
func DisposeOSLDisposeTokenUPP(OSLDisposeTokenUPP!)
```

Disposes of a universal procedure pointer to an object callback dispose token function.

```
func DisposeOSLGetErrDescUPP(OSLGetErrDescUPP!)
```

Disposes of a universal procedure pointer to an object callback get error descriptor function.

```
func DisposeOSLGetMarkTokenUPP(OSLGetMarkTokenUPP!)
```

Disposes of a universal procedure pointer to an object callback get mark function.

```
func DisposeOSLMarkUPP(OSLMarkUPP!)
```

Disposes of a universal procedure pointer to an object callback mark function.

```
func InvokeAECoerceDescUPP(UnsafePointer<AEDesc>!, DescType, SRefCon!,
UnsafeMutablePointer<AEDesc>!, AECoerceDescUPP!) -> OSErr
```

Calls a universal procedure pointer to a function that coerces data stored in a descriptor.

```
func InvokeAECoercePtrUPP(DescType, UnsafeRawPointer!, Size, DescType,
SRefCon!, UnsafeMutablePointer<AEDesc>!, AECoercePtrUPP!) -> OSErr
```

Calls a universal procedure pointer to a function that coerces data stored in a buffer.

```
func InvokeAEDisposeExternalUPP(UnsafeRawPointer!, Size, SRefCon!,
AEDisposeExternalUPP!)
```

Calls a dispose external universal procedure pointer.

```
func InvokeAEEventHandlerUPP(UnsafePointer<AppleEvent>!, UnsafeMutable
Pointer<AppleEvent>!, SRefCon!, AEEventHandlerUPP!) -> OSErr
```

Calls an event handler universal procedure pointer.

```
func InvokeOSLAccessorUPP(DescType, UnsafePointer<AEDesc>!, DescType,
DescType, UnsafePointer<AEDesc>!, UnsafeMutablePointer<AEDesc>!, SRef
Con!, OSLAccessorUPP!) -> OSErr
```

Calls an object accessor universal procedure pointer.

```
func InvokeOSLAdjustMarksUPP(Int, Int, UnsafePointer<AEDesc>!,
OSLAdjustMarksUPP!) -> OSErr
```

Calls an object callback adjust marks universal procedure pointer.

```
func InvokeOSLCompareUPP(DescType, UnsafePointer<AEDesc>!, Unsafe
Pointer<AEDesc>!, UnsafeMutablePointer<DarwinBoolean>!, OSLCompareUPP!)
-> OSErr
```

Calls an object callback comparison universal procedure pointer.

```
func InvokeOSLCountUPP(DescType, DescType, UnsafePointer<AEDesc>!,
UnsafeMutablePointer<Int>!, OSLCountUPP!) -> OSErr
```

Calls an object callback count universal procedure pointer.

```
func InvokeOSLDisposeTokenUPP(UnsafeMutablePointer<AEDesc>!, OSLDispose
TokenUPP!) -> OSErr
```

Calls an object callback dispose token universal procedure pointer.

```
func InvokeOSLGetErrDescUPP(UnsafeMutablePointer<UnsafeMutable
Pointer<AEDesc>?>!, OSLGetErrDescUPP!) -> OSErr
```

Calls an object callback get error descriptor universal procedure pointer.

```
func InvokeOSLGetMarkTokenUPP(UnsafePointer<AEDesc>!, DescType, Unsafe
MutablePointer<AEDesc>!, OSLGetMarkTokenUPP!) -> OSErr
```

Calls an object callback get mark universal procedure pointer.

```
func InvokeOSLMarkUPP(UnsafePointer<AEDesc>!, UnsafePointer<AEDesc>!,
Int, OSLMarkUPP!) -> OSErr
```

Calls an object callback mark universal procedure pointer.

```
func NewAECoerceDescUPP(AECoerceDescProcPtr!) -> AECoerceDescUPP!
```

Creates a new universal procedure pointer to a function that coerces data stored in a descriptor.

```
func NewAECoercePtrUPP(AECoercePtrProcPtr!) -> AECoercePtrUPP!
```

Creates a new universal procedure pointer to a function that coerces data stored in a buffer.

```
func NewAEDisposeExternalUPP(AEDisposeExternalProcPtr!) -> AEDispose
ExternalUPP!
```

Creates a new universal procedure pointer to a function that disposes of data stored in a buffer.

```
func NewAEEventHandlerUPP(AEEventHandlerProcPtr!) -> AEEventHandlerUPP!
```

Creates a new universal procedure pointer to an event handler function.

```
func NewOSLAccessorUPP(OSLAccessorProcPtr!) -> OSLAccessorUPP!
```

Creates a new universal procedure pointer to an object accessor function.

```
func NewOSLAdjustMarksUPP(OSLAdjustMarksProcPtr!) -> OSLAdjustMarksUPP!
```

Creates a new universal procedure pointer to an object callback adjust marks function.

```
func NewOSLCompareUPP(OSLCompareProcPtr!) -> OSLCompareUPP!
```

Creates a new universal procedure pointer to an object callback comparison function.

```
func NewOSLCountUPP(OSLCountProcPtr!) -> OSLCountUPP!
```

Creates a new universal procedure pointer to an object callback count function.

```
func NewOSLDisposeTokenUPP(OSLDisposeTokenProcPtr!) -> OSLDisposeToken
UPP!
```

Creates a new universal procedure pointer to an object callback dispose token function.

`func NewOSLGetErrDescUPP(OSLGetErrDescProcPtr!) -> OSLGetErrDescUPP!`

Creates a new universal procedure pointer to an object callback get error descriptor function.

`func NewOSLGetMarkTokenUPP(OSLGetMarkTokenProcPtr!) -> OSLGetMarkTokenUPP!`

Creates a new universal procedure pointer to an object callback get mark function.

`func NewOSLMarkUPP(OSLMarkProcPtr!) -> OSLMarkUPP!`

Creates a new universal procedure pointer to an object callback mark function.

Creating Descriptor Lists and Apple Event Records

`func AECreateList(UnsafeRawPointer!, Size, Bool, UnsafeMutablePointer<AEDescList>!) -> OSErr`

Creates an empty descriptor list or Apple event record.

Creating Object Specifiers

`func CreateCompDescriptor(DescType, UnsafeMutablePointer<AEDesc>!, UnsafeMutablePointer<AEDesc>!, Bool, UnsafeMutablePointer<AEDesc>!) -> OSErr`

Creates a comparison descriptor that specifies how to compare one or more Apple event objects with either another Apple event object or a descriptor.

`func CreateLogicalDescriptor(UnsafeMutablePointer<AEDescList>!, DescType, Bool, UnsafeMutablePointer<AEDesc>!) -> OSErr`

Creates a logical descriptor that specifies a logical operator and one or more logical terms for the Apple Event Manager to evaluate.

`func CreateObjSpecifier(DescType, UnsafeMutablePointer<AEDesc>!, DescType, UnsafeMutablePointer<AEDesc>!, Bool, UnsafeMutablePointer<AEDesc>!) -> OSErr`

Assembles an object specifier that identifies one or more Apple event objects, from other descriptors.

`func CreateOffsetDescriptor(Int, UnsafeMutablePointer<AEDesc>!) -> OSErr`

Creates an offset descriptor that specifies the position of an element in relation to the beginning or end of its container.

```
func CreateRangeDescriptor(UnsafeMutablePointer<AEDesc>!, UnsafeMutable
Pointer<AEDesc>!, Bool, UnsafeMutablePointer<AEDesc>!) -> OSErr
```

Creates a range descriptor that specifies a series of consecutive elements in the same container.

Deallocating Memory for Descriptors

```
func AEDisposeDesc(UnsafeMutablePointer<AEDesc>!) -> OSErr
```

Deallocates the memory used by a descriptor.

Deallocating Memory for Tokens

```
func AEDisposeToken(UnsafeMutablePointer<AEDesc>!) -> OSErr
```

Deallocates the memory used by a token.

Deleting Descriptors

```
func AEDeleteItem(UnsafeMutablePointer<AEDescList>!, Int) -> OSErr
```

Deletes a descriptor from a descriptor list, causing all subsequent descriptors to move up one place.

```
func AEDeleteParam(UnsafeMutablePointer<AppleEvent>!, AEKeyword) ->
OSErr
```

Deletes a keyword-specified parameter from an Apple event record.

Getting, Calling, and Removing Object Accessor Functions

```
func AECallObjectAccessor(DescType, UnsafePointer<AEDesc>!, DescType,
DescType, UnsafePointer<AEDesc>!, UnsafeMutablePointer<AEDesc>!) ->
OSErr
```

Invokes the appropriate object accessor function for a specific desired type and container type.

```
func AEGetObjectAccessor(DescType, DescType, UnsafeMutable
Pointer<OSLAccessorUPP?>!, UnsafeMutablePointer<SRefCon?>!, Bool) ->
OSErr
```

Gets an object accessor function from an object accessor dispatch table.

```
func AEInstallObjectAccessor(DescType, DescType, OSLAccessorUPP!, SRef
Con!, Bool) -> OSErr
```

Adds or replaces an entry for an object accessor function to an object accessor dispatch table.

```
func AERemoveObjectAccessor(DescType, DescType, OSLAccessorUPP!, Bool)
-> OSErr
```

Removes an object accessor function from an object accessor dispatch table.

Getting Data or Descriptors From Apple Events and Apple Event Records

```
func AEGetAttributeDesc(UnsafePointer<AppleEvent>!, AEKeyword, Desc
Type, UnsafeMutablePointer<AEDesc>!) -> OSErr
```

Gets a copy of the descriptor for a specified Apple event attribute from an Apple event; typically used when your application needs to pass the descriptor on to another function.

```
func AEGetAttributePtr(UnsafePointer<AppleEvent>!, AEKeyword, DescType,
UnsafeMutablePointer<DescType>!, UnsafeMutableRawPointer!, Size, Unsafe
MutablePointer<Size>!) -> OSErr
```

Gets a copy of the data for a specified Apple event attribute from an Apple event; typically used when your application needs to work with the data directly.

```
func AEGetParamDesc(UnsafePointer<AppleEvent>!, AEKeyword, DescType,
UnsafeMutablePointer<AEDesc>!) -> OSErr
```

Gets a copy of the descriptor for a keyword-specified Apple event parameter from an Apple event or an Apple event record.

```
func AEGetParamPtr(UnsafePointer<AppleEvent>!, AEKeyword, DescType,
UnsafeMutablePointer<DescType>!, UnsafeMutableRawPointer!, Size, Unsafe
MutablePointer<Size>!) -> OSErr
```

Gets a copy of the data for a specified Apple event parameter from an Apple event or an Apple event record.

Getting Information About the Apple Event Manager

```
func AEManagerInfo(AEKeyword, UnsafeMutablePointer<Int>!) -> OSErr
```

Provides information about the version of the Apple Event Manager currently available or the number of processes that are currently recording Apple events.

Getting Items From Descriptor Lists

```
func AEGetArray(UnsafePointer<AEDescList>!, AEArrayType, AEArrayData
Pointer!, Size, UnsafeMutablePointer<DescType>!, UnsafeMutable
Pointer<Size>!, UnsafeMutablePointer<Int>!) -> OSErr
```

Extracts data from an Apple event array created with the `AEPutArray` function and stores it as a standard array of fixed size items in the specified buffer.

```
func AEGetNthDesc(UnsafePointer<AEDescList>!, Int, DescType, Unsafe
MutablePointer<AEKeyword>!, UnsafeMutablePointer<AEDesc>!) -> OSErr
```

Copies a descriptor from a specified position in a descriptor list into a specified descriptor; typically used when your application needs to pass the extracted data to another function as a descriptor.

```
func AEGetNthPtr(UnsafePointer<AEDescList>!, Int, DescType, Unsafe
MutablePointer<AEKeyword>!, UnsafeMutablePointer<DescType>!, Unsafe
MutableRawPointer!, Size, UnsafeMutablePointer<Size>!) -> OSErr
```

Gets a copy of the data from a descriptor at a specified position in a descriptor list; typically used when your application needs to work with the extracted data directly.

Getting the Sizes and Descriptor Types of Descriptors

```
func AESizeOfAttribute(UnsafePointer<AppleEvent>!, AEKeyword, Unsafe
MutablePointer<DescType>!, UnsafeMutablePointer<Size>!) -> OSErr
```

Gets the size and descriptor type of an Apple event attribute from a descriptor of type `AppleEvent`.

```
func AESizeOfNthItem(UnsafePointer<AEDescList>!, Int, UnsafeMutable
Pointer<DescType>!, UnsafeMutablePointer<Size>!) -> OSErr
```

Gets the data size and descriptor type of the descriptor at a specified position in a descriptor list.

```
func AESizeOfParam(UnsafePointer<AppleEvent>!, AEKeyword, UnsafeMutable
Pointer<DescType>!, UnsafeMutablePointer<Size>!) -> OSErr
```

Gets the size and descriptor type of an Apple event parameter from a descriptor of type `AERecord` or `AppleEvent`.

Initializing the Object Support Library

```
func AEObjectInit() -> OSErr
```

Initializes the Object Support Library.

```
func AESetObjectCallbacks(OSLCompareUPP!, OSLCountUPP!, OSLDisposeTokenUPP!, OSLGetMarkTokenUPP!, OSLMarkUPP!, OSLAdjustMarksUPP!, OSLGetErrDescUPP!) -> OSErr
```

Specifies the object callback functions for your application.

Locating Processes on Remote Computers

Available starting in macOS version v10.3, these functions allow you to locate processes on remote computers (a task supported by the PPCToolbox in Mac OS 9).

```
func AECreateRemoteProcessResolver(CFAllocator!, CFURL!) -> AERemoteProcessResolverRef!
```

Creates an object for resolving a list of remote processes.

```
func AEDisposeRemoteProcessResolver(AERemoteProcessResolverRef!)
```

Disposes of an `AERemoteProcessResolverRef`.

```
func AERemoteProcessResolverGetProcesses(AERemoteProcessResolverRef!, UnsafeMutablePointer<CFStreamError>!) -> Unmanaged<CFArray>!
```

Returns an array of objects containing information about processes running on a remote machine.

```
func AERemoteProcessResolverScheduleWithRunLoop(AERemoteProcessResolverRef!, CFRunLoop!, CFString!, AERemoteProcessResolverCallback!, UnsafePointer<AERemoteProcessResolverContext>!)
```

Schedules a resolver for execution on a given run loop in a given mode.

Managing Apple Event Dispatch Tables

```
func AEGetEventHandler(AEEventClass, AEEventID, UnsafeMutablePointer<AEEventHandlerUPP?>!, UnsafeMutablePointer<SRefCon?>!, Bool) -> OSErr
```

Gets an event handler from an Apple event dispatch table.

```
func AEInstallEventHandler(AEEventClass, AEEventID, AEEventHandlerUPP!, SRefCon!, Bool) -> OSErr
```

Adds an entry for an event handler to an Apple event dispatch table.

```
func AERemoveEventHandler(AEEventClass, AEEventID, AEEventHandlerUPP!, Bool) -> OSErr
```

Removes an event handler entry from an Apple event dispatch table.

Managing Coercion Handler Dispatch Tables

```
func AEGetCoercionHandler(DescType, DescType, UnsafeMutable
Pointer<AECoercionHandlerUPP?>!, UnsafeMutablePointer<SRefCon?>!,
UnsafeMutablePointer<DarwinBoolean>!, Bool) -> OSErr
```

Gets the coercion handler for a specified descriptor type.

```
func AEInstallCoercionHandler(DescType, DescType, AECoercionHandler
UPP!, SRefCon!, Bool, Bool) -> OSErr
```

Installs a coercion handler in either the application or system coercion handler dispatch table.

```
func AERemoveCoercionHandler(DescType, DescType, AECoercionHandlerUPP!,
Bool) -> OSErr
```

Removes a coercion handler from a coercion handler dispatch table.

Managing Special Handler Dispatch Tables

```
func AEGetSpecialHandler(AEKeyword, UnsafeMutablePointer<AEEventHandler
UPP?>!, Bool) -> OSErr
```

Gets a specified handler from a special handler dispatch table.

```
func AEInstallSpecialHandler(AEKeyword, AEEventHandlerUPP!, Bool) ->
OSErr
```

Installs a callback function in a special handler dispatch table.

```
func AERemoveSpecialHandler(AEKeyword, AEEventHandlerUPP!, Bool) ->
OSErr
```

Removes a handler from a special handler dispatch table.

Operating On Descriptor Data

```
func AEGetDescData(UnsafePointer<AEDesc>!, UnsafeMutableRawPointer!,
Size) -> OSErr
```

Gets the data from the specified descriptor.

```
func AEGetDescDataSize(UnsafePointer<AEDesc>!) -> Size
```

Gets the size, in bytes, of the data in the specified descriptor.

```
func AEGetDescDataRange(UnsafePointer<AEDesc>!, UnsafeMutableRaw
Pointer!, Size, Size) -> OSStatus
```

Retrieves a specified series of bytes from the specified descriptor.

```
func AEReplaceDescData(DescType, UnsafeRawPointer!, Size, UnsafeMutable
Pointer<AEDesc>!) -> OSErr
```

Copies the specified data into the specified descriptor, replacing any previous data.

Resolving Object Specifiers

```
func AEResolve(UnsafePointer<AEDesc>!, Int16, UnsafeMutable
Pointer<AEDesc>!) -> OSErr
```

Resolves an object specifier.

Creating Apple Event Structures in Memory

```
func AEPrintDescToHandle(UnsafePointer<AEDesc>!, UnsafeMutable
Pointer<Handle?>!) -> OSStatus
```

Provides a pretty printer facility for displaying the contents of Apple event descriptors.

```
func vAEBuildAppleEvent(AEEventClass, AEEventID, DescType, UnsafeRaw
Pointer!, Size, Int16, Int32, UnsafeMutablePointer<AppleEvent>!, Unsafe
MutablePointer<AEBuildError>!, UnsafePointer<CChar>!, CVaListPointer) -
> OSStatus
```

Allows you to encapsulate calls to `AEBuildAppleEvent` in a wrapper routine.

```
func vAEBuildDesc(UnsafeMutablePointer<AEDesc>!, UnsafeMutable
Pointer<AEBuildError>!, UnsafePointer<CChar>!, CVaListPointer) ->
OSStatus
```

Allows you to encapsulate calls to `AEBuildDesc` in your own wrapper routines.

```
func vAEBuildParameters(UnsafeMutablePointer<AppleEvent>!, Unsafe
MutablePointer<AEBuildError>!, UnsafePointer<CChar>!, CVaListPointer) -
> OSStatus
```

Allows you to encapsulate calls to `AEBuildParameters` in your own `stdarg`-style wrapper routines, using techniques similar to those allowed by vsprintf.

Creating Apple Event Structures Using Streams

```
func AEStreamClose(AEStreamRef!, UnsafeMutablePointer<AEDesc>!) ->
OSStatus
```

Closes and deallocates an `AEStreamRef`.

```
func AEStreamCloseDesc(AEStreamRef!) -> OSStatus
```

Marks the end of a descriptor in an `AEStreamRef`.

```
func AEStreamCloseList(AEStreamRef!) -> OSStatus
```

Marks the end of a list of descriptors in an `AEStreamRef`.

```
func AEStreamCloseRecord(AEStreamRef!) -> OSStatus
```

Marks the end of a record in an `AEStreamRef`.

```
func AEStreamCreateEvent(AEEventClass, AEEventID, DescType, UnsafeRaw
Pointer!, Size, Int16, Int32) -> AEStreamRef!
```

Creates a new Apple event and opens a stream for writing data to it.

```
func AEStreamOpen() -> AEStreamRef!
```

Opens a new `AEStreamRef` for use in building a descriptor.

```
func AEStreamOpenDesc(AEStreamRef!, DescType) -> OSStatus
```

Marks the beginning of a descriptor in an `AEStreamRef`.

```
func AEStreamOpenEvent(UnsafeMutablePointer<AppleEvent>!) -> AEStream
Ref!
```

Opens a stream for an existing Apple event.

```
func AEStreamOpenKeyDesc(AEStreamRef!, AEKeyword, DescType) -> OSStatus
```

Marks the beginning of a key descriptor in an `AEStreamRef`.

```
func AEStreamOpenList(AEStreamRef!) -> OSStatus
```

Marks the beginning of a descriptor list in an `AEStreamRef`.

```
func AEStreamOpenRecord(AEStreamRef!, DescType) -> OSStatus
```

Marks the beginning of an Apple event record in an `AEStreamRef`.

```
func AEStreamOptionalParam(AEStreamRef!, AEKeyword) -> OSStatus
```

Designates a parameter in an Apple event as optional.

```
func AEStreamSetRecordType(AEStreamRef!, DescType) -> OSStatus
```

Sets the type of the most recently created record in an `AEStreamRef`.

```
func AEStreamWriteAEDesc(AEStreamRef!, UnsafePointer<AEDesc>!) ->
OSStatus
```

Copies an existing descriptor into an `AEStreamRef`.

```
func AEStreamWriteData(AEStreamRef!, UnsafeRawPointer!, Size) ->
OSStatus
```

Appends data to the current descriptor in an `AEStreamRef`.

```
func AEStreamWriteDesc(AEStreamRef!, DescType, UnsafeRawPointer!, Size)
-> OSStatus
```

Appends the data for a complete descriptor to an `AEStreamRef`.

```
func AEStreamWriteKey(AEStreamRef!, AEKeyword) -> OSStatus
```

Marks the beginning of a keyword/descriptor pair for a descriptor in an `AEStreamRef`.

```
func AEStreamWriteKeyDesc(AEStreamRef!, AEKeyword, DescType, UnsafeRaw
Pointer!, Size) -> OSStatus
```

Writes a complete keyword/descriptor pair to an `AEStreamRef`.

Working With Lower Level Apple Event Functions

```
func AEGetRegisteredMachPort() -> mach_port_t
```

Returns the Mach port (in the form of a `mach_port_t`) that was registered with the bootstrap server for this process.

```
func AEDecodeMessage(UnsafeMutablePointer<mach_msg_header_t>!, Unsafe
MutablePointer<AppleEvent>!, UnsafeMutablePointer<AppleEvent>!) ->
OSStatus
```

Decodes a Mach message and converts it into an Apple event and its related reply.

```
func AESendMessage(UnsafePointer<AppleEvent>!, UnsafeMutable
Pointer<AppleEvent>!, AESendMode, Int) -> OSStatus
```

Sends an AppleEvent to a target process without some of the overhead required by `AESend`.

```
func AEProcessMessage(UnsafeMutablePointer<mach_msg_header_t>!) ->
OSStatus
```

Decodes and dispatches a low level Mach message event to an event handler, including packaging and returning the reply to the sender.

Serializing Apple Event Data

```
func AESizeOfFlattenedDesc(UnsafePointer<AEDesc>!) -> Size
```

Returns the amount of buffer space needed to store the descriptor after flattening it.

`func AEFlattenDesc(UnsafePointer<AEDesc>!, Ptr!, Size, UnsafeMutablePointer<Size>!) -> OSStatus`

Flattens the specified descriptor and stores the data in the supplied buffer.

~~`func AEUnflattenDesc(UnsafeRawPointer!, UnsafeMutablePointer<AEDesc>!) -> OSStatus`~~

Unflattens the data in the passed buffer and creates a descriptor from it.

Deprecated

Miscellaneous

`func AECheckIsRecord(UnsafePointer<AEDesc>!) -> Bool`

Determines whether a descriptor is truly an `AERecord`.

`func AEInitializeDesc(UnsafeMutablePointer<AEDesc>!)`

Initializes a new descriptor.

Callbacks

`typealias AERemoteProcessResolverCallback`

Defines a pointer to a function the Apple Event Manager calls when the asynchronous execution of a remote process resolver completes, either due to success or failure, after a call to the `AERemoteProcessResolverScheduleWithRunLoop` function. Your callback function can use the reference passed to it to get the remote process information.

`typealias AEDisposeExternalProcPtr`

Defines a pointer to a function the Apple Event Manager calls to dispose of a descriptor created by the `AECreateDescFromExternalPtr` function. Your callback function disposes of the buffer you originally passed to that function.

`typealias AECoerceDescProcPtr`

Defines a pointer to a function that coerces data stored in a descriptor. Your descriptor coercion callback function coerces the data from the passed descriptor to the specified type, returning the coerced data in a second descriptor.

`typealias AECoercePtrProcPtr`

Defines a pointer to a function that coerces data stored in a buffer. Your pointer coercion callback routine coerces the data from the passed buffer to the specified type, returning the coerced data in a descriptor.

`typealias AEEventHandlerProcPtr`

Defines a pointer to a function that handles one or more Apple events. Your Apple event handler function performs any action requested by the Apple event, adds parameters to the reply Apple event if appropriate (possibly including error information), and returns a result code.

`typealias` `OSLAccessorProcPtr`

Your object accessor function either finds elements or properties of an Apple event object.

`typealias` `OSLAdjustMarksProcPtr`

Defines a pointer to an adjust marks callback function. Your adjust marks function unmarks objects previously marked by a call to your marking function.

`typealias` `OSLCompareProcPtr`

Defines a pointer to an object comparison callback function. Your object comparison function compares one Apple event object to another or to the data for a descriptor.

`typealias` `OSLCountProcPtr`

Defines a pointer to an object counting callback function. Your object counting function counts the number of Apple event objects of a specified class in a specified container object.

`typealias` `OSLDisposeTokenProcPtr`

Defines a pointer to a dispose token callback function. Your dispose token function, required only if you use a complex token format, disposes of the specified token.

`typealias` `OSLGetErrDescProcPtr`

Defines a pointer to an error descriptor callback function. Your error descriptor callback function supplies a pointer to an address where the Apple Event Manager can store the current descriptor if an error occurs during a call to the `AEResolve` function.

`typealias` `OSLGetMarkTokenProcPtr`

Defines a pointer to a mark token callback function. Your mark token function returns a mark token.

`typealias` `OSLMarkProcPtr`

Defines a pointer to an object marking callback function. Your object-marking function marks a specific Apple event object.

Data Types

`struct` `AEArrayData`

Stores array information to be put into a descriptor listwith the `AEPutArray` functionor extracted from a descriptor list with the `AEGetArray` function.

`struct` `AEBuildError`

Defines a structure for storing additional error codeinformation for “AEBuild” routines.

`struct` `AEDesc`

Stores data and an accompanying descriptor type to formthe basic building block of all Apple Events.

`struct` `AEKeyDesc`

Associates a keyword with a descriptor to form a keyword-specifieddescriptor.

`struct` `AERemoteProcessResolverContext`

Supplied as a parameter when performing asynchronous resolutionof remote processes.

`struct` `ccntTokenRecord`

Stores token information used by the AEResolve functionwhile locating a range of objects.

`struct` `IntlText`

International text consists of an ordered series of bytes, beginning with a 4-byte language code and a 4-byte script code that together determine the format of the bytes that follow.

`struct` `OffsetArray`

Specifies offsets of ranges of text. Not typically used by developers.

`struct` `TextRange`

Specifies a range of text. Not typically used by developers.

`struct` `TextRangeArray`

Specifies an array of text ranges. Not typically used by developers.

`struct` `TScriptingSizeResource`

Defines a data type to store stack and heap information. Not typically used by developers.

`struct` `WritingCode`

`typealias` `AEAddressDesc`

A descriptor that contains the address of an application, used to describe the target application for an Apple event.

`typealias` `AEArrayDataPointer`

A pointer to a union of type `AEArrayData`.

`typealias` `AEArrayType`

Stores a value that specifies an array type.

`typealias` `AECoerceDescUPP`

Defines a data type for the universal procedure pointer for the `AECoerceDescProcPtr` callback function pointer.

`typealias` `AECoercePtrUPP`

Defines a data type for the universal procedure pointer for the `AECoercePtrProcPtr` callback function pointer.

`typealias` `AECoercionHandlerUPP`

Defines a data type for the universal procedure pointer for the `AECoercionHandlerUPP` callback function pointer.

`typealias` `AEDataStorage`

A pointer to an opaque data type that provides storage for an `AEDesc` descriptor.

`typealias` `AEDataStorageType`

An opaque data type used to store data in Apple event descriptors.

`typealias` `AEDescList`

A descriptor whose data consists of a list of one or more descriptors.

`typealias` `AEEventSource`

A data type for values that specify how an Apple event was delivered.

`typealias` `AEDisposeExternalUPP`

Defines a universal procedure pointer to a function the Apple Event Manager calls to dispose of a descriptor created by the `AECreateDescFromExternalPtr` function.

`typealias` `AEEventClass`

Specifies the event class of an Apple event.

`typealias` `AEEventHandlerUPP`

Defines a data type for the universal procedure pointer for the `AEEventHandlerUPP` callback function pointer.

`typealias` `AEEventID`

Specifies the event ID of an Apple event.

`typealias` `AEKeyword`

A four-character code that uniquely identifies a descriptor in an Apple event record or an Apple event.

`typealias` `AERecord`

A descriptor whose data is a list of keyword-specified descriptors.

`typealias` `AERemoteProcessResolverRef`

An opaque reference to an object that encapsulates the mechanism for obtaining a list of processes running on a remote machine.

`typealias` `AEReturnID`

Specifies a return ID for a created Apple event.

`typealias` `AESendPriority`

Specifies the processing priority for a sent Apple event.

`typealias` `AEStreamRef`

An opaque data structure for storing stream-based descriptor data.

`typealias` `AETransactionID`

Specifies a transaction ID.

`typealias` `AppleEvent`

A descriptor whose data is a list of descriptors containing both attributes and parameters that make up an Apple event.

`typealias` `DescType`

Specifies the type of the data stored in an `AEDesc` descriptor.

`typealias` `OffsetArrayHandle`

Defines a data type that points to an `OffsetArray`. Not typically used by developers.

`typealias` `OSLAccessorUPP`

Defines a data type for the universal procedure pointer for the `OSLAccessorProcPtr` callback function pointer.

`typealias` `OSLAdjustMarksUPP`

Defines a data type for the universal procedure pointer for the `OSLAdjustMarksProcPtr` callback function pointer.

`typealias` `OSLCompareUPP`

Defines a data type for the universal procedure pointer for the `OSLCompareProcPtr` callback function pointer.

`typealias` `OSLCountUPP`

Defines a data type for the universal procedure pointer for the `OSLCountProcPtr` callback function pointer.

`typealias` `OSLDisposeTokenUPP`

Defines a data type for the universal procedure pointer for the `OSLDisposeTokenProcPtr` callback function pointer.

`typealias` `OSLGetErrDescUPP`

Defines a data type for the universal procedure pointer for the `OSLGetErrDescProcPtr` callback function pointer.

`typealias` `OSLGetMarkTokenUPP`

Defines a data type for the universal procedure pointer for the `OSLGetMarkTokenProcPtr` callback function pointer.

`typealias` `OSLMarkUPP`

Defines a data type for the universal procedure pointer for the `OSLMarkProcPtr` callback function pointer.

Constants

`typealias` `AEBuildErrorCode`

Represents syntax errors found by an Apple Event build routine.

`typealias` `AESendMode`

Specify send preferences to the `AESend` function.

Apple Event Recording Event ID Constants

Specify event IDs for events that deal with Apple event recording.

cAEList

Callback Constants for the AEResolve Function

Specify supported callback features to the `AEResolve` function.

cInsertionLoc

cKeystroke

`Comparison Operator Constants`

Specify a comparison operation to perform on two operands.

Constants for Object Specifiers, Positions, and Logical and Comparison Operations

Specify the types of the four keyword-specified descriptors that make up the data in an object specifier, as well as constants for position, logical operations, and comparison operations.

- cURL

`cVersion`

- Data Array Constants

  Specify an array type for storing or extracting descriptor lists with the `AEPutArray` and `AEGetArray` functions.

- Descriptor Type Constants

  Specify types for descriptors.

- eScheme

- Event Class Constants

  Specify the event class for an Apple event.

- Event ID Constants

  Specify the event ID for an Apple event.

- Event Source Constants

  Identify how an Apple event was delivered.

- Factoring Constants

- ID Constants for the AECreateAppleEvent Function

  Specify values for the ID parameters of the `AECreateAppleEvent` function.

- Key Form and Descriptor Type Object Specifier Constants

  Specify possible values for the `keyAEKeyForm` field of an object specifier, as well as descriptor types used in resolving object specifiers.

- Keyword Attribute Constants

  Specify keyword values for Apple event attributes.

- Keyword Parameter Constants

  Specify keyword values for Apple event parameters, as well as information for the `AEManagerInfo` function to retrieve. Some common key word values are shown here.

- Launch Apple Event Constants

In a `kAEOpenApplication` event, specify information about how the receiving application was launched.

- Numeric Descriptor Type Constants
  Specify types for numeric descriptors.
- Object Class ID Constants
  Specify the object class for an Apple event object.
- Other Descriptor Type Constants
  Specify types for Boolean and character descriptors.
- Priority Constants for the AESend Function (Deprecated in macOS)
  Specify a value for the `sendPriority` parameter of the `AESend` function.
- Remote Process Dictionary Keys
  Used to extract information from dictionaries with entries that describe remote processes.
- Special Handler Callback Constants
  Specify an object callback function to install, get, or remove from the special handler dispatch table.
- Timeout Constants
  Specify a timeout value.

`Whose Test Constants`

`kAEDoObjectsExist`

- kAEDebugPOSTHeader

`kAEGetPrivilegeSelection`

- kAEHandleArray
- kAEInfo
- kAEInternetSuite
- kAEISGetURL
- kAEISHTTPSearchArgs
- kAELogOut
- kAEMenuClass
- kAEMouseClass

- kAENonmodifiable
- kAEQDNotOr
- kAESetPosition
- kAESocks4Protocol
- kAEUseHTTPProxyAttr

  Web Services Proxy support—these constants should be added as attributes of the event that is being sent (not as part of the direct object).
- kAEUserTerminology
- kAEUseSocksAttr
- kAEUTHasReturningParam
- kAEZoomIn
- kBySmallIcon
- kConnSuite
- keyAEAngle
- keyAEBaseAddr
- keyAEDoScale
- keyAEHiliteRange
- keyAEKeyword

`keyAEPropData`

- keyAESuiteID
- keyMenuID
- keyMiscellaneous
- keyReplyPortAttr
- keySOAPStructureMetaData
- keyUserNameAttr
- kFAServerApp
- kLaunchToGetTerminology

- kNextBody
- kOSIZDontOpenResourceFile
- kReadExtensionTermsMask
- kSOAP1999Schema
- kTextServiceClass
- kTSMHiliteCaretPosition
  Specify text highlighting information.
- kTSMOutsideOfBody
- pArcAngle
- pFormula
- pNewElementLoc
- pScheme
- pTextStyles
- typeAEText
- typeApplicationBundleID
  For specifying a target application by bundle ID.

`typeFinderWindow`

- typeHIMenu
- typeKernelProcessID
  For specifying an application by UNIX process ID.

`typeMachPort`
: For specifying a Mach port.

- typeMeters

`typePixelMap`

- typeReplyPortAttr
- typeTIFF
- typeUnicodeText

Result Codes

Because the Apple Event Manager uses the services of the EventManager, the functions described in this document may return EventManager result codes in addition to the Apple Event Manager resultcodes listed here. Less commonly, an Apple Event Manager functionmay return other result codes, including some of those found inthe CarbonCore header file `MacErrors.h`.

For result codes for the AEBuild-related functions, see `AEBuildErrorCode`.

`var noPortErr: Int`

Client hasn't set `'SIZE'` resource toindicate awareness of high-level events

`var destPortErr: Int`

Server hasn't set `'SIZE'` resource toindicate awareness of high-level events, or else is not present

`var sessClosedErr: Int`

The `kAEDontReconnect` flagin the `sendMode` parameterwas set and the server quit, then restarted

`var errAECoercionFail: Int`

Data could not be coerced to the requesteddescriptor type

`var errAEDescNotFound: Int`

Descriptor was not found

`var errAECorruptData: Int`

Data in an Apple event could not be read

`var errAEWrongDataType: Int`

Wrong descriptor type

`var errAENotAEDesc: Int`

Not a valid descriptor

`var errAEBadListItem: Int`

Operation involving a list item failed

`var errAENewerVersion: Int`

Need a newer version of the Apple EventManager

`var errAENotAppleEvent: Int`

The event is not in AppleEvent format.

`var errAEEventNotHandled: Int`

Event wasn’t handled by an Apple eventhandler

`var errAEReplyNotValid: Int`

`AEResetTimer` was passed an invalid reply

`var errAEUnknownSendMode: Int`

Invalid sending mode was passed

`var errAEWaitCanceled: Int`

User canceled out of wait loop for replyor receipt

`var errAETimeout: Int`

Apple event timed out

`var errAENoUserInteraction: Int`

No user interaction allowed

`var errAENotASpecialFunction: Int`

Wrong keyword for a special function

`var errAEParamMissed: Int`

A required parameter was not accessed.

`var errAEUnknownAddressType: Int`

Unknown Apple event address type

`var errAEHandlerNotFound: Int`

No handler found for an Apple event

`var errAEReplyNotArrived: Int`

Reply has not yet arrived

`var errAEIllegalIndex: Int`

Not a valid list index

`var errAEImpossibleRange: Int`

The range is not valid because it is impossiblefor a range to include the first and last objects that were specified;an example is a range in which the offset of the first object is greaterthan the offset of the last object

```
var errAEWrongNumberArgs: Int
```

The number of operands provided for the kAENOT logicaloperator is not 1

```
var errAEAccessorNotFound: Int
```

There is no object accessor function forthe specified object class and container type

```
var errAENoSuchLogical: Int
```

The logical operator in a logical descriptoris not kAEAND, kAEOR,or kAENOT

```
var errAEBadTestKey: Int
```

The descriptor in a test key is neithera comparison descriptor nor a logical descriptor

```
var errAENoSuchObject: Int
```

Runtime resolution of an object failed.

```
var errAENegativeCount: Int
```

An object-counting function returned a negativeresult

```
var errAEEmptyListContainer: Int
```

The container for an Apple event objectis specified by an empty list

```
var errAEUnknownObjectType: Int
```

The object type isn’t recognized

```
var errAERecordingIsAlreadyOn: Int
```

Recording is already on

```
var errAEReceiveTerminate: Int
```

Break out of all levels of AEReceive tothe topmost (1.1 or greater)

```
var errAEReceiveEscapeCurrent: Int
```

Break out of lowest level only of AEReceive (1.1or greater)

```
var errAEEventFiltered: Int
```

Event has been filtered and should not bepropagated (1.1 or greater)

```
var errAEDuplicateHandler: Int
```

Attempt to install handler in table foridentical class and ID (1.1 or greater)

```
var errAEStreamBadNesting: Int
```

Nesting violation while streaming

```
var errAEStreamAlreadyConverted: Int
```

Attempt to convert a stream that has alreadybeen converted

```
var errAEDescIsNull: Int
```

Attempt to perform an invalid operationon a null descriptor

```
var errAEBuildSyntaxError: Int
```

`AEBuildDesc` andrelated functions detected a syntax error

```
var errAEBufferTooSmall: Int
```

Buffer for `AEFlattenDesc` toosmall

```
var errASCantConsiderAndIgnore: Int
```

Can’t both consider and ignore <attribute>.

```
var errASCantCompareMoreThan32k: Int
```

Can’t perform operation on text longerthan 32K bytes.

```
var errASTerminologyNestingTooDeep: Int
```

Tell statements are nested too deeply.

```
var errASIllegalFormalParameter: Int
```

<name> is illegal as a formal parameter.

```
var errASParameterNotForEvent: Int
```

<name> is not a parameter name for the event <event>.

```
var errASNoResultReturned: Int
```

No result was returned for some argumentof this expression.

```
var errAEEventFailed: Int
```

Apple event handler failed.

```
var errAETypeError: Int
```

A descriptor type mismatch occurred.

```
var errAEBadKeyForm: Int
```

Invalid key form.

```
var errAENotModifiable: Int
```

Can't set <object or data> to <object or data>. Access not allowed.

```
var errAEPrivilegeError: Int
```

A privilege violation occurred.

```
var errAEReadDenied: Int
```

The read operation was not allowed.

```
var errAEWriteDenied: Int
```

Can't set <object or data> to <object or data>.

```
var errAEIndexTooLarge: Int
```

The index of the event is too large to bevalid.

```
var errAENotAnElement: Int
```

The specified object is a property, notan element.

```
var errAECantSupplyType: Int
```

Can’t supply the requested descriptortype for the data.

```
var errAECantHandleClass: Int
```

The Apple event handler can’t handle objectsof this class.

```
var errAEInTransaction: Int
```

Couldn’t handle this command because itwasn’t part of the current transaction.

```
var errAENoSuchTransaction: Int
```

The transaction to which this command belongedisn’t a valid transaction.

```
var errAENoUserSelection: Int
```

There is no user selection.

```
var errAENotASingleObject: Int
```

Handler only handles single objects.

```
var errAECantUndo: Int
```

Can’t undo the previous Apple event oruser action.

```
var errAENotAnEnumMember: Int
```

Enumerated value in `SetData` is notallowed for this property

```
var errAECantPutThatThere: Int
```

In make new, duplicate, etc. class can'tbe an element of container

```
var errAEPropertiesClash: Int
```

Illegal combination of properties settingsfor SetData, make new, or duplicate

---

See Also

Managers

- ColorSync Manager
- Speech Synthesis Manager