HIObject Reference

Carbon > User Experience

2005-08-11

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

Simultaneously published in the United States and Canada.

Contents

HIObject Reference 7

Overview 7
Functions by Task 8
- Registering and Creating HIObjects 8
- HIObject Utility Functions 8
- Accessibility Functions 9
- Archiving Functions 9
- Miscellaneous Functions 9

Functions 9
- HIObjectCopyClassID 9
- HIObjectCopyCustomArchiveData 10
- HIObjectCreate 10
- HIObjectCreateFromBundle 11
- HIObjectDynamicCast 12
- HIObjectGetEventTarget 12
- HIObjectIsAccessibilityIgnored 13
- HIObjectIsArchivingIgnored 13
- HIObjectIsOfClass 13
- HIObjectOverrideAccessibilityContainment 14
- HIObjectPrintDebugInfo 15
- HIObjectRegisterSubclass 15
- HIObjectSetAccessibilityIgnored 17
- HIObjectSetArchivingIgnored 17
- HIObjectSetAuxiliaryAccessibilityAttribute 18
- HIObjectSetCustomArchiveData 19
- HIObjectUnregisterClass 20

Constants 20
- Standard Custom Archive Data Dictionary Keys for Custom Initialize Events 20
- Standard Custom Archive Data Dictionary Class and SuperClass Keys 21
- Standard Custom Archive Data Dictionary Key for ProcPointer-Based CDEFs 21
- HIObject Base Class Events 21
- HIObject Base Class Event Parameters 23

Result Codes 25

Document Revision History 27

Index 29

Tables

HIObject Reference 7

Table 1 Parameter names and types for HIObject base class events 23

HIObject Reference

Framework:	Carbon/Carbon.h
Declared in	HIAccessibility.h HIObject.h

Overview

HIObject is the base class for various objects in the HIToolbox. In Mac OS X v10.2 and later, most common user interface objects (controls, windows, menus, toolbars, and toolbar items) are derived from HIObject. Code that is external to HIToolbox can also create its own subclasses of these objects using the routines contained in this API. There are also polymorphic functions one can use on any HIObject for getting the class ID, and so on.

HIObjects are actually Core Foundation CF types under the hood. This means that they can be put into CF collections and you can retain/release them.

An HIObject is essentially a very basic building-block object which contains an event target. You can create these objects to use as your own Carbon Event receptors in your application, or you can subclass existing HIToolbox objects to suit your needs.

You register your subclasses with `HIObjectRegisterSubclass` (page 15), passing your class ID, the parent class, and an event handler. You also pass a list of events the handler is interested in.

To create an object of your subclass, you call `HIObjectCreate` (page 10), passing the class reference you registered, as well as an initialization event.

Construction is two-phase: first the basic construction of the object is done, then initialization is performed. The HIToolbox sends construction events bottom-up, as you would expect in C++ or the like. Here is the list of what goes on to create an object:

1) The HIToolbox creates the base HIObject

2) It then installs the event handler you specified when you registered your subclass. Your handler must listen for `kEventHIObjectConstruct` and `kEventHIObjectDestruct` events. If it does not, the class cannot be registered (you will get a `paramErr`).

3) Next, the HIToolbox directly calls your handler with an `kEventHIObjectConstruct` event. When called like this, you are not really being called in the context of a handler stack, so you cannot do things like `CallNextEventHandler`. The `userData` parameter is what you specified when you registered the class. Typically, during construction you will allocate memory yourself to store your own instance data; this allocation might be as simple as calling `malloc` or `NewPtr`, or it might involve creating your own C++ object. In the construct event, you are passed the base `HIObjectRef` of the object being created. Typically you would

store this `HIObjectRef` in your own instance data for later use. When handling this construct event, you should be sure to use `SetEventParameter` to set the `kEventParamHIObjectInstance` parameter in the construction event with your own instance data. You must use `typeVoidPtr` as the type.

4) The HIToolbox looks for your instance of `typeVoidPtr` after you handle the construct event. It then takes that data and stores it off with the object and also sets the user data of the event handler it installed to be this instance data. This means that following the construct event, all calls to your event handler will have the instance data you returned to us.

5) Once construction has completed successfully, we will send your object the initialize event passed into `HIObjectCreate`. At this point, all events are now sent to your object using standard Carbon event mechanisms (it is only the construct event which is special). When we send the initialization event to your subclass, you should pass the event to your superclass before proceeding. You do this with `CallNextEventHandler`. Once back from that call, you should verify that the result is noErr, indicating that the superclass did in fact initialize properly. If it did not, your should return the error that `CallNextEventHandler` returned from your handler as well. The object will be destroyed by the HIToolbox. Your object should be able to be destroyed in a partially initialized state such as this. This stage is optional, i.e. an object does not need to respond to the initialize event unless it is expecting certain parameters to be passed to it at creation time. This is where those parameters can be fetched.

6) Once initialization is successful, the `HIObjectRef` is returned to the caller of `HIObjectCreate`.

When someone has called `CFRelease` enough such that the reference count of the object drops to zero, the object is destroyed. The HIToolbox will send a `kEventHIObjectDestruct` event to your object. Do not call `CallNextEventHandler`. Just clean up and return from your handler.

For more information about HIObjects and the HIView subclass, see *HIView Programming Guide*.

Functions by Task

Registering and Creating HIObjects

`HIObjectRegisterSubclass` (page 15)
Registers an HIObject subclass.

`HIObjectCreate` (page 10)
Creates an object derived from HIObject.

`HIObjectCreateFromBundle` (page 11)
Obtains the HIObject for the given bundle.

`HIObjectUnregisterClass` (page 20)
Unregisters a previously registered subclass of HIObject.

HIObject Utility Functions

`HIObjectCopyClassID` (page 9)
Obtains the class ID of a given HIObject.

`HIObjectIsOfClass` (page 13)
Determines whether an object is of a certain class.

`HIObjectDynamicCast` (page 12)
Obtains the instance data for a specific class of an HIObject.

`HIObjectGetEventTarget` (page 12)
Obtains the event target of an `HIObjectRef`.

Accessibility Functions

`HIObjectSetAccessibilityIgnored` (page 17)
Marks an HIObject as ignored (or not) for the purposes of the accessibility APIs.

`HIObjectIsAccessibilityIgnored` (page 13)
Reports whether the given HIObject is marked as ignored for accessibility.

`HIObjectSetAuxiliaryAccessibilityAttribute` (page 18)
Associates an additional accessibility attribute with an HIObject.

`HIObjectOverrideAccessibilityContainment` (page 14)
Overrides the AXUIElement references supplied by an HIObject.

Archiving Functions

`HIObjectIsArchivingIgnored` (page 13)
Obtains a Boolean value indicating whether an HIObject is marked as ignored for archiving.

`HIObjectSetArchivingIgnored` (page 17)
Changes the state of archiving for an HIObject.

`HIObjectCopyCustomArchiveData` (page 10)
Copies custom archive data that is associated with an HIObject.

`HIObjectSetCustomArchiveData` (page 19)
Associates custom archive data with an HIObject.

Miscellaneous Functions

`HIObjectPrintDebugInfo` (page 15)
Prints the internal information of an HIObject for debugging purposes.

Functions

HIObjectCopyClassID

Obtains the class ID of a given HIObject.

```
CFStringRef HIObjectCopyClassID (
   HIObjectRef inObject
);
```

Parameters

inObject

The object whose class ID you want.

Return Value
A reference to the object's class ID.

Availability
Available in Mac OS X v10.2 and later.

Declared In
HIObject.h

HIObjectCopyCustomArchiveData

Copies custom archive data that is associated with an HIObject.

```
OSStatus HIObjectCopyCustomArchiveData (
   HIObjectRef inObject,
   CFDictionaryRef *outCustomData
);
```

Parameters

inObject

The HIObject whose custom archive data you want to retrieve.

outCustomData

On return, a pointer to the custom data, or NULL if no custom archive data is associated with the specified object. The caller is responsible for releasing the dictionary when it is no longer needed.

Return Value
An operating system result code.

Discussion
This function would be used by an archive editor to get custom archive data associated with an HIObject so that the data can be edited.

Availability
Available in Mac OS X v10.4 and later.
Not available to 64-bit applications.

Declared In
HIObject.h

HIObjectCreate

Creates an object derived from HIObject.

```
OSStatus HIObjectCreate (
   CFStringRef inClassID,
   EventRef inConstructData,
   HIObjectRef *outObject
);
```

Parameters

inClassID

The class ID of the class of object you want to instantiate.

inConstructData

If your class (or any class you derive from) accepts creation parameters, you need to pass an event into this parameter. The class must be `kEventClassHIObject`, and the kind should be `kEventHIObjectInitialize`. Any other parameters should be added as necessary. Specific subclasses of HIObject which require initialization parameters will specify those parameters in the appropriate headers.

outObject

The instance of the object you create.

Return Value

A result code. See "HIObject Result Codes" (page 25).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`HIObject.h`

HIObjectCreateFromBundle

Obtains the HIObject for the given bundle.

```
OSStatus HIObjectCreateFromBundle (
   CFBundleRef inBundle,
   HIObjectRef *outObject
);
```

Parameters

inBundle

The bundle with which you want to communicate.

outObject

The HIObject associated with the bundle.

Return Value

A result code. See "HIObject Result Codes" (page 25). If the bundle's HIObject creation function cannot be found, `cfragNoSymbolErr` will be returned.

Discussion

A bundle can be designed to communicate with an application through an HIObject. The bundle must be designed to create an HIObject and have a defined suite of Carbon Events that clients can use to communicate with the bundle's HIObject. Given a CFBundleRef, this API will tell the bundle to create the HIObject and return it to the caller.

Availability

Available in Mac OS X v10.2 and later.

Declared In
HIObject.h

HIObjectDynamicCast

Obtains the instance data for a specific class of an HIObject.

```
void * HIObjectDynamicCast (
   HIObjectRef inObject,
   CFStringRef inClassID
);
```

Parameters

inObject

The object whose class ID you want to check.

inClassID

The class ID to get the instance data for.

Return Value
A void * result containing the instance data for the object, or NULL if the object is not an instance of the class.

Discussion
The instance data returned is the same instance data the class's construction event handler returns in the instance data parameter. This is stored off with the class reference so that it can be fetched later for use by this function. It allows your subclass to easily get at the data it created, if your subclass needs that data outside of an event handler. (Inside an event handler, your subclass can get at its instance data via the userData parameter to the event handler.)

Availability
Available in Mac OS X v10.2 and later.

Declared In
HIObject.h

HIObjectGetEventTarget

Obtains the event target of an HIObjectRef.

```
EventTargetRef HIObjectGetEventTarget (
   HIObjectRef inObject
);
```

Parameters

inObject

The object whose target you want.

Return Value
An EventTargetRef.

Availability
Available in Mac OS X v10.2 and later.

Declared In
HIObject.h

HIObjectIsAccessibilityIgnored

Reports whether the given HIObject is marked as ignored for accessibility.

```
Boolean HIObjectIsAccessibilityIgnored (
   HIObjectRef inObject
);
```

Parameters

inObject
The object whose accessibility ignored state you want to query.

Return Value

A Boolean whose value is `true` if the object is marked as ignored or `false` if the object is not marked as ignored.

Discussion

See the discussion of `HIObjectSetAccessibilityIgnored` (page 17) for details on what it means to be accessibility ignored.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIAccessibility.h`

HIObjectIsArchivingIgnored

Obtains a Boolean value indicating whether an HIObject is marked as ignored for archiving.

```
Boolean HIObjectIsArchivingIgnored (
   HIObjectRef inObject
);
```

Parameters

inObject
The HIObject whose archiving-ignored state is to be queried.

Return Value

A Boolean whose value is `true` if the specified HIObject is ignored for archiving; otherwise, `false`.

Discussion

By default, HIObjects are marked as ignored for archiving, which indicates that the HIObject does not support the archiving protocol.

Availability

Available in Mac OS X v10.4.

Declared In

`HIObject.h`

HIObjectIsOfClass

Determines whether an object is of a certain class.

```
Boolean HIObjectIsOfClass (
   HIObjectRef inObject,
   CFStringRef inObjectClassID
);
```

Parameters

inObject

The object whose class ID you want to check.

inObjectClassID

The class ID in question.

Return Value
A Boolean whose value is `true` if the object is of the specified class; otherwise, `false`.

Discussion
You can use this to see whether an object you have derives from an expected superclass.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`HIObject.h`

HIObjectOverrideAccessibilityContainment

Overrides the AXUIElement references supplied by an HIObject.

```
OSStatus HIObjectOverrideAccessibilityContainment (
   HIObjectRef inHIObject,
   AXUIElementRef inDesiredParent,
   AXUIElementRef inDesiredWindow,
   AXUIElementRef inDesiredTopLevelUIElement
);
```

Parameters

inHIObject

The object whose parent attribute you want to override.

inDesiredParent

The UI element value you want the HIObject to supply for the parent attribute. This function makes a copy of the `AXUIElementRef`, so you must release `inDesiredParent` when this function returns. Pass `NULL` if you want the specified HIObject to supply its normal parent.

inDesiredWindow

The UI element value you want the HIObject to supply for the window attribute. This function makes a copy of the `AXUIElementRef`, so you must release this parameter when this function returns. Pass `NULL` if you want the specified HIObject to supply its normal window, if any.

inDesiredTopLevelUIElement

The UI element you want the HIObject to supply for the top-level element attribute. This function makes a copy of the `AXUIElementRef`, so you must release this parameter when this function returns. Passing `NULL` indicates that you want the HIObject to supply is normal top-level element, if any.

Return Value
An operating system result code.

Discussion
Use this function to change the parent that an HIObject would normally supply in the accessibility hierarchy. For example, a pop-up control would call this function on its menu so that the menu supplies the pop-up control as the menu's parent instead of the application, which would normally be supplied as the menu's parent. You can also use this function to change the window and the top-level element that an HIObject would normally supply.

If the specified HIObject is a standard HIToolbox construct, for example, an HIView or a menu, the specified HIObject is not added as an accessibility child of its normal parent. If it is not a standard construct, the caller is responsible for ensuring that the specified HIObject is not added as an accessibility child of its normal parent.

If the desired `AXUIElementRef` parent represents an HIView, menu, or window, the specified HIObject is automatically added as an accessibility child of the specified parent. In all other cases, it is the caller's responsibility to add the specified HIObject manually as a child of the specified parent. To represent an HIView, menu, or window, an `AXUIElementRef` object must contain the appropriate `HIObjectRef` value and an identifier value of `0`.

Currently, containment overrides are only supported by HIObjects that are of type HIView, Menu, or Window.

Availability
Available in Mac OS X v10.4 and later.
Not available to 64-bit applications.

Declared In
`HIAccessibility.h`

HIObjectPrintDebugInfo

Prints the internal information of an HIObject for debugging purposes.

```
void HIObjectPrintDebugInfo (
   HIObjectRef inObject
);
```

Parameters
`inObject`
The object to inspect.

Discussion
This function sends the information to `stdout`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`HIObject.h`

HIObjectRegisterSubclass

Registers an HIObject subclass.

```
OSStatus HIObjectRegisterSubclass (
   CFStringRef inClassID,
   CFStringRef inBaseClassID,
   OptionBits inOptions,
   EventHandlerUPP inConstructProc,
   ItemCount inNumEvents,
   const EventTypeSpec *inEventList,
   void *inConstructData,
   HIObjectClassRef *outClassRef
);
```

Parameters

inClassID

The class ID of your class. It should be unique. We recommend using Java-style com.company.foo naming conventions to avoid collisions.

inBaseClassID

The class ID of the class you derive from. Passing `NULL` indicates you want to subclass HIObject (the base class) directly.

inOptions

Any special options for your class. Currently you must pass `0` for this parameter.

inConstructProc

A universal procedure pointer to the event handler for this subclass. You pass the address of an event handler into this parameter. This handler is called directly, rather than through the normal event-dispatching mechanism. This means that the EventHandlerCallRef passed in will be `NULL`, and you cannot use it for calls like `CallNextEventHandler`. Other than that, you should return a result as usual. After your object is constructed, this procedure is installed as the event handler for the remaining events specified in the `inEventList` parameter. In Mac OS X v10.4 and later, passing `NULL` creates an "abstract class" that cannot be instantiated but can still be used as a base class for subclasses. If you pass `NULL`, `HIObjectCreate` on the class ID will return `hiObjectClassIsAbstractErr`.

inNumEvents

The number of events you are installing.

inEventList

The events your handler wants to receive. You must handle the `kEventHIObjectConstruct` and `kEventHIObjectDestruct` event. If these events are not specified, an error is returned.

inConstructData

Pass any info you want passed into your event handler here. For a C++ hierarchy based on HIObjects, you might actually pass a static method to construct your object here, and the base class event handler to do construction as your event handler.

outClassRef

The newly created class reference. Pass `NULL` if you don't care.

Return Value

A result code. See "HIObject Result Codes" (page 25).

Availability

Available in Mac OS X version 10.2 (v10.2) and later.

Declared In

`HIObject.h`

HIObjectSetAccessibilityIgnored

Marks an HIObject as ignored (or not) for the purposes of the accessibility APIs.

```
OSStatus HIObjectSetAccessibilityIgnored (
   HIObjectRef inObject,
   Boolean inIgnored
);
```

Parameters

inObject

The object whose accessibility ignored state you want to change.

inIgnored

A Boolean whose value is `true` to mark the object as ignored or `false` to mark the object as not ignored.

Return Value
An operating system result code.

Discussion
An HIObject that is ignored for accessibility will never be shown to an assistive application that uses the accessibility APIs to examine an interface. Your application's accessibility implementation can (and should) still report an ignored HIObject as usual. Carbon's accessibility engine will automatically prune any ignored HIObjects out of the data that is shown to an assistive application.

By default, an HIObject is *not* accessibility-ignored.

Availability
Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In
`HIAccessibility.h`

HIObjectSetArchivingIgnored

Changes the state of archiving for an HIObject.

```
OSStatus HIObjectSetArchivingIgnored (
   HIObjectRef inObject,
   Boolean inIgnored
);
```

Parameters

inObject

The HIObject whose archiving ignored state is to be changed.

inIgnored

A Boolean whose value is `true` to indicate that the specified HIObject does not support the archiving protocol and should be ignored; otherwise, `false`.

Return Value
An operating system result code.

Discussion
Call this function to mark or unmark an HIObject as ignored for archiving. By default, HIObjects are marked as ignored for archiving. HIObject subclasses supporting archiving with the `kEventHIObjectInitialize` and `kEventHIObjectEncode` events must set their archiving ignored state to `false` in order to receive archiving requests from clients. A client may still reset the archive ignored state to `true` on a particular object. An HIObject marked as ignored for archiving will never be requested to encode itself into an archive.

Availability
Available in Mac OS X v10.4 and later.
Not available to 64-bit applications.

Declared In
`HIObject.h`

HIObjectSetAuxiliaryAccessibilityAttribute

Associates an additional accessibility attribute with an HIObject.

```
OSStatus HIObjectSetAuxiliaryAccessibilityAttribute (
   HIObjectRef inHIObject,
   UInt64 inIdentifier,
   CFStringRef inAttributeName,
   CFTypeRef inAttributeData
);
```

Parameters

inHIObject
The part of the object-identifier pair that is to be associated with attribute data.

inIdentifier
The part of the object-identifier pair with which the attribute data is to be associated. Pass `0` if you want to associate the attribute data to the HIObject as a whole. For example, pass `0` if you want to associate a description attribute to a push button.

inAttributeName
The name of the attribute that is to be associated with the object-identifier pair. This string is retained before it is added to the auxiliary attribute store.

inAttributeData
The data that is to become the attribute's value. This data is retained before it is added to the auxiliary attribute store. You should release `inAttributeData` after `HIObjectSetAuxiliaryAccessibilityAttribute` returns. Pass `NULL` to indicate that the named auxiliary attribute should no longer be associated with the object-identifier pair. When you pass `NULL`, any named attribute data that was previously associated with the object-identifier pair is released.

Return Value
An operating system result code.

Discussion
Use this function to provide the name and data for an accessibility attribute you want to add to the UIElement representing an HIObject-identifier pair. Normally, Carbon events are used to supply accessibility attributes dynamically, but `HIObjectSetAuxiliaryAccessibilityAttribute` allows you to supply them statically.

This function is particularly useful for supplying values for the `kAXDescriptionAttribute`, `kAXTitleUIElementAttribute`, `kAXServesAsTitleUIElementAttribute`, `kAXLinkedUIElementsAttribute`, and other attributes whose values are specific to the layout and usage

of your application. Note, however, that this function can associate only attributes whose values do not change. If you need to supply attributes whose values are determined dynamically or whose values can be set, you must install the normal accessibility Carbon event handlers for normal accessibility.

When an accessibility attribute Carbon event is handled by the HIObject with a given identifier, the HIToolbox automatically supplies the names and values of any auxiliary attributes associated with the HIObject-identifier pair.

The auxiliary attribute store is consulted during the HIObject's default handling of accessibility attribute Carbon events. This means that any programmatic handling of a given accessibility attribute is able to override or block consultation of the store. In general, if the HIToolbox or a Carbon event handler can provide the attribute value in some other way, the store is not consulted.

Availability
Available in Mac OS X v10.4 and later.
Not available to 64-bit applications.

Declared In
`HIAccessibility.h`

HIObjectSetCustomArchiveData

Associates custom archive data with an HIObject.

```
OSStatus HIObjectSetCustomArchiveData (
   HIObjectRef inObject,
   CFDictionaryRef inCustomData
);
```

Parameters

inObject
The HIObject with which custom archive data is to be associated.

inCustomData
A `CFDictionaryRef` containing the custom archive data that is to be associated with the specified HIObject. Associating the custom archive data replaces any data that was previously associated. To archive, the dictionary's keys and values must use CFType callbacks. Pass `NULL` to clear any custom archive data that was previously associated.

Return Value
An operating system result code.

Discussion
This function might be used by an archive editor to associate custom archive data that it has edited with an HIObject.

Availability
Available in Mac OS X v10.4 and later.
Not available to 64-bit applications.

Declared In
`HIObject.h`

HIObjectUnregisterClass

Unregisters a previously registered subclass of HIObject.

```
OSStatus HIObjectUnregisterClass (
   HIObjectClassRef inClassRef
);
```

Parameters

inClassRef

The class reference of the object class you want to unregister.

Discussion

You receive an error if there are subclasses of your class or existing instances of it. All instances and subclasses must be disposed of and unregistered first.

Availability

Available in Mac OS X v10.2 and later.

Declared In

HIObject.h

Constants

Standard Custom Archive Data Dictionary Keys for Custom Initialize Events

Define standard custom archive dictionary keys for custom initialize events.

```
const CFStringRef kHIObjectCustomDataParameterNamesKey;
const CFStringRef kHIObjectCustomDataParameterTypesKey;
const CFStringRef kHIObjectCustomDataParameterValuesKey;
```

Constants

`kHIObjectCustomDataParameterNamesKey`

The value of this key is an array of strings. Each `CFStringRef` contains an `OSType` that is a Carbon event parameter name.

`kHIObjectCustomDataParameterTypesKey`

The value of this key is an array of strings. Each `CFStringRef` contains an `OSType` that is a Carbon event parameter type.

`kHIObjectCustomDataParameterValuesKey`

The value of this key is an array of strings. Each `CFStringRef` contains a representation of the value.

Discussion

The value for each of these constants is a `CFArrayRef` containing CFStringRefs. For a given dictionary, the names, types, and values arrays should each have the same number of CFStrings. The name-type-value triple at the given index in each array represents a Carbon event parameter in the initialization event for the HIObject.

The current supported set of Carbon event parameter types consists of `'cfst'`, `'TEXT'`, `'bool'`, `'cfrn'`, `'doub'`, `'osst'`, `'long'`, `'magn'`, `'hipt'`, `'hisz'`, and `'hirc'`.

Availability

Available in Mac OS v10.4 and later.

Standard Custom Archive Data Dictionary Class and SuperClass Keys

Define standard custom archive data dictionary keys for classes and superclasses.

```
const CFStringRef kHIObjectCustomDataClassIDKey;
const CFStringRef kHIObjectCustomDataSuperClassIDKey;
```

Constants

`kHIObjectCustomDataClassIDKey`
The class ID.

`kHIObjectCustomDataSuperClassIDKey`
The super-class ID key.

Discussion

These keys define a class and superclass for clients that do not implement the object's true class. Each keyed value is an HIObject class ID.

Availability

Available in Mac OS v10.4 and later.

Standard Custom Archive Data Dictionary Key for ProcPointer-Based CDEFs

Define a standard custom archive data dictionary key for ProcPointer-based CDEFs.

```
const CFStringRef kHIObjectCustomDataCDEFProcIDKey;
```

Constants

`kHIObjectCustomDataCDEFProcIDKey`
The standard custom archive data dictionary key for ProcPointer-based CDEFs.

Discussion

The key value is a `CFStringRef`-based signed 16-bit integer. Use `CFStringGetIntValue` in `CFString.h` to convert `CFStringRef` to `SInt16` and to convert `SInt16` to `CFStringRef`.

Availability

Available in Mac OS v10.4 and later.

HIObject Base Class Events

Define the base-class functionality of HIObjects.

```
enum{
    kEventClassHIObject = 'hiob',
    kEventHIObjectConstruct = 1,
    kEventHIObjectInitialize = 2,
    kEventHIObjectDestruct = 3,
    kEventHIObjectIsEqual = 4,
    kEventHIObjectPrintDebugInfo = 5,
    kEventHIObjectEncode = 6
};
```

Constants

`kEventClassHIObject`

The event class for HIObject events.

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventHIObjectConstruct`

Your object is being constructed. When your event handler is called with this event, it is being called directly and not through the normal event dispatching mechanism. This means that the `EventHandlerCallRef` passed to your handler is `NULL` and `CallNextEventHandler` does not work. You are passed the actual `HIObjectRef` of your base class for you to record in your instance data. (Available in Mac OS X v10.2 and later.)

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventHIObjectInitialize`

Your object is being initialized. Your handler should pass this onto the superclass first before handling this event. This is done by calling `CallNextEventHandler` with the event. When that function returns, you should make sure the result is `noErr`. If not, you should not continue to initialize your object. (Available in Mac OS X v10.2 and later.)

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventHIObjectDestruct`

Your object is being destroyed. This is your chance to dispose of anything you might have allocated for your object. Do not call through with `CallNextEventHandler`. (Available in Mac OS X v10.2 and later.)

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventHIObjectIsEqual`

`HIObjectIsEqual` has been called, and you are being asked to determine whether your object is equivalent to the one being passed to your handler. If your object is equivalent, you should place `true` in the `kEventParamResult` parameter in the event; if your object is not equivalent, place `false` in the `kEventParamResult` parameter. (Available in Mac OS X v10.2 and later.)

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventHIObjectPrintDebugInfo`
`HIObjectPrintDebugInfo` has been called, and you are being asked to print your information to stdout. This event is sent to all handlers and you should not call `CallNextEventHandler`. (Available in Mac OS X v10.2 and later.)

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventHIObjectEncode`
`HIArchiveEncodeHIObject` has been called on your HIObject, and you are being asked to encode your object into an archive. Before handling this event, your handler should pass this event to the superclass by calling `CallNextEventHandler` with the event. If `CallNextEventHandler` does not return `noErr`, you should not continue to encode your instance data. (Available in Mac OS X v10.4 and later.)

Available in Mac OS X v10.4 and later.

Declared in `HIObject.h`.

Discussion
You only need to be aware of these events if you are implementing a subclass.

Table 1 Parameter names and types for HIObject base class events

Event kind	Parameter name	Parameter type
`kEventHIObjectConstruct`	`kEventParamHIObjectInstance`	`typeHIObjectRef`
`kEventHIObjectInitialize`	`kEventParamHIArchive`	`typeCFTypeRef`
`kEventHIObjectDestruct`		
`kEventHIObjectIsEqual`	`kEventParamDirectObject`	`typeHIObjectRef`
	`kEventParamResult`	`typeBoolean`
`kEventHIObjectPrintDebugInfo`		
`kEventHIObjectEncode`	`kEventParamHIArchive`	`typeCFTypeRef`

HIObject Base Class Event Parameters

Define constants for HIObject base class event parameters.

```
enum {
    kEventParamHIObjectInstance = 'hioi',
    kEventParamHIArchive = 'hiac',
    typeHIObjectRef = 'hiob'
};
```

Constants

`kEventParamHIObjectInstance`

On entry, the HIObject reference for your object. This parameter is currently only used for the `kEventHIObjectConstruct` event. Typically, you read this parameter from the event and store it in your instance data so that when your instance needs to call HIObject APIs, your instance can use this `HIObjectRef`. On exit, the value of this parameter is `typeVoidPtr` and is a pointer to the instance data you have written into the event using `SetEventParameter`. After your event handler returns, the HIToolbox reads your instance data pointer from the event, and installs the event handlers that were passed to `HIObjectRegisterClassSubclass` on the new object. The HIToolbox uses the instance data pointer as the refcon for the event handlers it installed so that your event handlers can retrieve your instance data pointer from the reference constant (the third parameter to an `EventHandlerProcPtr`).(Available in Mac OS X v10.2 and later.)

Available in Mac OS X v10.2 and later.

Declared in `HIObject.h`.

`kEventParamHIArchive`

An HIArchive used to store or retrieve the HIObject. This parameter is passed in the following cases:

- In the `kEventHIObjectInitialize` event when the HIObject is requested to initialize itself from a decoded archive. If the HIObject is to be initialized normally (that is, not from an archive), the initialize event does not contain `kEventParamHIArchive` .
- In the `kEventHIObjectEncode` event, which is sent to request that the HIObject encode itself within an archive.

(Available in Mac OS X v10.4 and later.)

Available in Mac OS X v10.4 and later.

Declared in `HIObject.h`.

Result Codes

Result Code	Value	Description
`hiObjectClassExistsErr`	-22080	You are trying to register a class ID that already exists. Available in Mac OS X v10.2 and later.
`hiObjectClassHasInstancesErr`	-22081	You are trying to unregister a class which has instances that still exist. Available in Mac OS X v10.2 and later.
`hiObjectClassHasSubclassesErr`	-22082	You are trying to unregister a class which has subclasses registered. They must be unregistered before this class can be unregistered. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
`hiObjectClassIsAbstractErr`	-22083	You are trying to create an HIObject class that is defined as being abstract. You must subclass it instead. Available in Mac OS X v10.2 and later.

Document Revision History

This table describes the changes to *HIObject Reference*.

Date	Notes
2005-08-11	Added updates and corrections. Fixed formatting bug.
2005-06-04	Updated for Mac OS X v10.4.
2003-05-01	Conversion from latest Jaguar HeaderDoc

Document Revision History

Index

H

HIObject Base Class Event Parameters 23
HIObject Base Class Events 21
hiObjectClassExistsErr constant 24
hiObjectClassHasInstancesErr constant 24
hiObjectClassHasSubclassesErr constant 24
hiObjectClassIsAbstractErr constant 25
HIObjectCopyClassID function 9
HIObjectCopyCustomArchiveData function 10
HIObjectCreate function 10
HIObjectCreateFromBundle function 11
HIObjectDynamicCast function 12
HIObjectGetEventTarget function 12
HIObjectIsAccessibilityIgnored function 13
HIObjectIsArchivingIgnored function 13
HIObjectIsOfClass function 13
HIObjectOverrideAccessibilityContainment function 14
HIObjectPrintDebugInfo function 15
HIObjectRegisterSubclass function 15
HIObjectSetAccessibilityIgnored function 17
HIObjectSetArchivingIgnored function 17
HIObjectSetAuxiliaryAccessibilityAttribute function 18
HIObjectSetCustomArchiveData function 19
HIObjectUnregisterClass function 20

K

kEventClassHIObject constant 22
kEventHIObjectConstruct constant 22
kEventHIObjectDestruct constant 22
kEventHIObjectEncode constant 23
kEventHIObjectInitialize constant 22
kEventHIObjectIsEqual constant 22
kEventHIObjectPrintDebugInfo constant 23
kEventParamHIArchive constant 24
kEventParamHIObjectInstance constant 24
kHIObjectCustomDataCDEFProcIDKey constant 21
kHIObjectCustomDataClassIDKey constant 21
kHIObjectCustomDataParameterNamesKey constant 20
kHIObjectCustomDataParameterTypesKey constant 20
kHIObjectCustomDataParameterValuesKey constant 20
kHIObjectCustomDataSuperClassIDKey constant 21

S

Standard Custom Archive Data Dictionary Class and SuperClass Keys 21
Standard Custom Archive Data Dictionary Key for ProcPointer-Based CDEFs 21
Standard Custom Archive Data Dictionary Keys for Custom Initialize Events 20