HIView Reference

Carbon > User Experience

2007-04-13

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

Simultaneously published in the United States and Canada.

Contents

Tables

HIView Reference 11

HIView Reference

Framework:	Carbon/Carbon.h
Companion guide	HIView Programming Guide
Declared in	HIClockView.h HICocoaView.h HIComboBox.h HIImageViews.h HIMenuView.h HIObject.h HIScrollView.h HISearchField.h HISegmentedView.h HITextViews.h HIToolboxDebugging.h HIView.h HIWindowViews.h Menus.h

Overview

HIView is an object-oriented view system subclassed from HIObject. All controls are implemented as HIView objects ("views"). You can easily subclass HIView classes, making it easy to implement custom controls. Over time, the HIView API will replace the current Control Manager. Using the HIView model, every item within a window is a view: the root control, controls, and even the standard window "widgets" (close, zoom, and minimize buttons, resize control, and so on). Current Control Manager function calls are layered on top of this HIView model.

Additional benefits of the HIView model include the following:

- Quartz is the native drawing system, but you can still use QuickDraw if desired.
- Modern coordinate system not bounded by the 16-bit space of QuickDraw.
- Simplified coordinate system for view bounds and the position of a view within its parent.
- Views can be ordered within a hierarchy layer; that is, it is easy to place controls in front of or behind other controls.

For additional information about using HIViews, see *HIView Programming Guide*.

Functions by Task

Obtaining and Placing Views

`HIViewGetRoot` (page 75)
Obtains the root view for a window.

`HIViewFindByID` (page 64)
Obtains a view by its ID.

`HIViewGetSuperview` (page 77)
Returns a view's parent view.

`HIViewPlaceInSuperviewAt` (page 86)
Places a view at an absolute location within its parent.

`HIViewGetNextView` (page 73)
Returns the view behind the specified view.

`HIViewGetPreviousView` (page 75)
Returns the view above the specified view.

Working With Subviews

`HIViewAddSubview` (page 54)
Adds a subview to the given parent view.

`HIViewRemoveFromSuperview` (page 87)
Removes a view from its parent.

`HIViewGetFirstSubview` (page 68)
Returns the first subview of a parent view.

`HIViewGetLastSubview` (page 71)
Returns the last subview in a parent view.

`HIViewCountSubviews` (page 62)
Returns the number of subviews embedded in a view.

`HIViewGetIndexedSubview` (page 70)
Obtains the subview of a view by index.

Manipulating Views

`HIViewSetVisible` (page 103)
Hides or shows a view.

`HIViewIsVisible` (page 84)
Determines whether a view is visible.

`HIViewIsLatentlyVisible` (page 82)
Determines whether a view is latently visible.

`HIViewSetHilite` (page 94)
Sets highlighting on a view.

`HIViewIsActive` (page 80)
Determines whether a view is active.

`HIViewSetActivated` (page 90)
Sets a view to be active or inactive.

`HIViewIsEnabled` (page 82)
Determines whether a view is enabled.

`HIViewSetEnabled` (page 92)
Enables or disables a view.

`HIViewIsCompositingEnabled` (page 81)
Determines whether compositing is enabled for a view.

`HIViewSetText` (page 101)
Sets the text of a view to the specified string.

`HIViewCopyText` (page 61)
Copies the text of a view.

`HIViewGetValue` (page 78)
Obtains the value of a view.

`HIViewSetValue` (page 102)
Sets the value of a view.

`HIViewGetMinimum` (page 72)
Obtains the minimum value of a view.

`HIViewSetMinimum` (page 97)
Sets a view's minimum value.

`HIViewGetMaximum` (page 72)
Obtains a view's maximum value.

`HIViewSetMaximum` (page 97)
Sets a view's maximum value.

`HIViewGetViewSize` (page 79)
Obtains the view size of a view.

`HIViewSetViewSize` (page 102)
Sets the view size of a view.

`HIViewIsValid` (page 84)
Determines whether the specified view is known to the HIToolbox.

`HIViewGetID` (page 69)
Obtains the HIViewID of a view.

`HIViewSetID` (page 95)
Sets the `HIViewID` of a view.

`HIViewGetCommandID` (page 66)
Obtains the command ID of a view.

`HIViewSetCommandID` (page 91)
Sets the command ID of a view.

`HIViewGetKind` (page 70)
Obtains the signature and kind of a view.

`HIViewGetAttributes` (page 65)
Obtains the attributes for a view.

`HIViewChangeAttributes` (page 56)
Changes the attributes of a view.

`HIViewGetNeedsDisplay` (page 72)
Determines whether a view needs to be redrawn.

`HIViewSetNeedsDisplay` (page 98)
Marks a view as needing or not needing to be redrawn.

`HIViewSetNeedsDisplayInRect` (page 98)
Uses an `HIRect` to mark a portion of a view as needing or not needing to be redrawn.

`HIViewSetNeedsDisplayInShape` (page 100)
Uses a shape to mark a portion of a view as needing or not needing to be redrawn.

`HIViewSetNeedsDisplayInRegion` (page 99)
Uses a region to mark a portion of a view as needing or not needing to be redrawn.

`HIViewRender` (page 88)
Renders the invalid portions of a view.

`HIViewGetSizeConstraints` (page 76)
Returns the minimum and maximum size for a control.

`HIViewIsDrawingEnabled` (page 81)
Determines if drawing is currently enabled for a view.

`HIViewSetDrawingEnabled` (page 92)
Turns control drawing on or off.

`HIViewScrollRect` (page 89)
Scrolls a view's contents, or a portion thereof.

`HIViewSetZOrder` (page 103)
Changes the front-to-back ordering of sibling views.

`HIViewReshapeStructure` (page 88)
Informs the system that the structure region of the given view has changed shape.

`HIViewRegionChanged` (page 87)
Informs the system that a region of the view has changed.

`HIViewCopyShape` (page 60)
Copies the shape of a part of a view.

`HIViewGetOptimalBounds` (page 73)
Obtains the optimal size and text placement of a view.

`HIViewFlashDirtyArea` (page 65)
Flashes a window's dirty area.

`HIViewGetWindow` (page 80)
Obtains a reference to the window to which the specified view is bound.

`HIViewGetFeatures` (page 67)
Obtains the features of the specified view.

`HIViewChangeFeatures` (page 57)
Changes the features of a view.

`HIViewGetEventTarget` (page 67)
Returns the `EventTargetRef` for the specified view.

Managing Focus

`HIViewGetFocusPart` (page 68)
Obtains the part in the specified view that currently has focus.

`HIViewSetNextFocus` (page 100)
Sets the view that is to receive keyboard focus when keyboard focus advances from the specified view.

`HIViewAdvanceFocus` (page 55)
Advances the keyboard focus to the next most appropriate view.

`HIViewSubtreeContainsFocus` (page 105)
Determines whether a view or any subviews have keyboard focus.

`HIViewSetFirstSubViewFocus` (page 93)
Sets the subview that is first to receive keyboard focus.

Processing Events and Hit-Testing for Views

`HIViewClick` (page 58)
Passes a mouse-down event to a view.

`HIViewSimulateClick` (page 104)
Simulates a mouse click on a given view.

`HIViewGetPartHit` (page 74)
Determines the part hit for a given point.

`HIViewGetViewForMouseEvent` (page 78)
Returns the appropriate view to handle a mouse event.

`HIViewGetSubviewHit` (page 76)
Returns the child of the given view hit by the point passed in.

Manipulating View Coordinates

`HIViewGetBounds` (page 65)
Obtains the local bounds of a view.

`HIViewSetBoundsOrigin` (page 91)
Sets the origin of the view.

`HIViewGetFrame` (page 68)
Obtains the frame bounds of a view.

`HIViewSetFrame` (page 94)
Sets the frame of a view.

`HIViewMoveBy` (page 85)
Move a view by the specified distance relative to its current location.

`HIViewConvertPoint` (page 59)
Converts a point's coordinates from one view to another.

`HIViewConvertRect` (page 59)
Converts a rectangle from one view to another.

HIViewConvertRegion (page 60)
Converts a region from one view to another.

Creating and Manipulating Combo Boxes

HIComboBoxCreate (page 24)
Creates a combo box control.

HIComboBoxGetAttributes (page 24)
Gets the attributes of a combo box.

HIComboBoxChangeAttributes (page 22)
Changes the attributes of a combo box.

HIComboBoxAppendTextItem (page 22)
Appends a text item to the combo box disclosure list.

HIComboBoxCopyTextItemAtIndex (page 23)
Copy a text item from a combo box disclosure list

HIComboBoxGetItemCount (page 25)
Gets the number of items in the combo box disclosure list.

HIComboBoxInsertTextItemAtIndex (page 25)
Inserts a CFString in a combo box disclosure list.

HIComboBoxRemoveItemAtIndex (page 26)
Removes an item from a combo box disclosure list.

HIComboBoxIsListVisible (page 26)
Determines whether a combo box disclosure list is visible.

HIComboBoxSetListVisible (page 27)
Hides or shows a combo box disclosure list.

Creating and Manipulating Image Views

HIImageViewCreate (page 29)
Creates an image view.

HIImageViewCopyImage (page 29)
Obtains the image for an image view.

HIImageViewSetImage (page 32)
Sets the image to display in an image view.

HIImageViewGetAlpha (page 30)
Obtains the alpha value for a view.

HIImageViewSetAlpha (page 31)
Sets the alpha value for an image view.

HIImageViewGetScaleToFit (page 30)
Determines whether an image will scale or clip to the view bounds.

HIImageViewSetScaleToFit (page 33)
Specifies whether an image should scale or clip to the view's bounds.

HIImageViewIsOpaque (page 31)
Determines whether an image view is opaque.

HIImageViewSetOpaque (page 32)
Sets the opacity of an image view.

Creating and Manipulating Scroll Views

HIScrollViewCreate (page 35)
Creates a scroll view.

HIScrollViewGetScrollBarAutoHide (page 36)
Obtains current setting of a scroll view's scroll bar auto-hide setting.

HIScrollViewSetScrollBarAutoHide (page 37)
Sets a scroll view's auto-hide setting.

HIScrollViewCanNavigate (page 35)
Determines whether it is possible to navigate in a scroll view.

HIScrollViewNavigate (page 37)
Changes the portion of a view's target.

Creating and Manipulating Layouts

HIViewGetLayoutInfo (page 71)
Obtains the layout information of an view.

HIViewSetLayoutInfo (page 95)
Sets the layout information of an HIView.

HIViewApplyLayout (page 56)
Applies the current layout to the specified view.

HIViewSuspendLayout (page 105)
Suspends layout handling for a view and its children.

HIViewResumeLayout (page 89)
Resumes layout handling for a view and its children.

HIViewIsLayoutActive (page 83)
Determines whether layout handling is active or suspended.

HIViewIsLayoutLatentlyActive (page 83)
Determines whether layout handling is latently active or suspended.

Manipulating Tracking Areas

HIViewNewTrackingArea (page 85)
Creates a new tracking area for a view.

HIViewChangeTrackingArea (page 58)
Changes the shape of a tracking area.

HIViewGetTrackingAreaID (page 77)
Obtains the ID of a tracking area.

HIViewDisposeTrackingArea (page 63)
Disposes of an existing tracking area.

Creating and Manipulating Search Fields

HISearchFieldCreate (page 39)
Creates a Search field control.

HISearchFieldSetSearchMenu (page 41)
Sets the search menu associated with a search field view.

HISearchFieldGetSearchMenu (page 40)
Obtains the search menu associated with a search field.

HISearchFieldChangeAttributes (page 38)
Sets the attributes of a search field.

HISearchFieldGetAttributes (page 40)
Obtains the attributes for a search field.

HISearchFieldSetDescriptiveText (page 41)
Sets the description of the search action for a search field.

HISearchFieldCopyDescriptiveText (page 38)
Obtains the description associated with a search field.

Manipulating Menus

HIMenuViewGetMenu (page 34)
Returns the MenuRef associated with a view that is a subclass of HIMenuView.

HIMenuGetContentView (page 33)
Obtains an HIViewRef that can be used to draw menu content for a menu.

Manipulating Segmented Views

HISegmentedViewCreate (page 44)
Creates a segmented view.

HISegmentedViewSetSegmentCount (page 51)
Sets the number of segments for a segmented view.

HISegmentedViewGetSegmentCount (page 47)
Obtains the number of segments for a segmented view.

HISegmentedViewSetSegmentBehavior (page 49)
Changes the behavior of an individual segment of a segmented view.

HISegmentedViewGetSegmentBehavior (page 45)
Obtains the behavior of an individual segment of a segmented view.

HISegmentedViewChangeSegmentAttributes (page 42)
Changes the attributes of an individual segment of a segmented view.

HISegmentedViewGetSegmentAttributes (page 45)
Returns the attributes of an individual segment of a segmented view.

HISegmentedViewSetSegmentValue (page 54)
Changes the value of an individual segment of a segmented view.

HISegmentedViewGetSegmentValue (page 48)
Returns the value of an individual segment of a segmented view.

HISegmentedViewSetSegmentEnabled (page 52)
Enables or disables an individual segment of a segmented view.

HISegmentedViewIsSegmentEnabled (page 49)
Determines whether an individual segment of a segmented view is enabled.

HISegmentedViewSetSegmentCommand (page 50)
Sets the command ID for a segment.

HISegmentedViewGetSegmentCommand (page 46)
Obtains the command ID associated with a segment.

HISegmentedViewSetSegmentLabel (page 53)
Sets the label string for a segment.

HISegmentedViewCopySegmentLabel (page 43)
Obtains a copy of the label string associated with a segment.

HISegmentedViewSetSegmentContentWidth (page 50)
Specifies how the content width of segment is to be calculated.

HISegmentedViewGetSegmentContentWidth (page 46)
Obtains the content width of a segment.

HISegmentedViewSetSegmentImage (page 52)
Sets or clears the image associated with a segment.

HISegmentedViewGetSegmentImageContentType (page 48)
Obtains the type of image content drawn by a segment.

HISegmentedViewCopySegmentImage (page 43)
Copies the image drawn by a segment.

Working with Core Graphics Images

HICreateTransformedCGImage (page 27)
Creates a new Core Graphics image with the standard selected or disabled appearance.

HIViewCreateOffscreenImage (page 62)
Creates a Core Graphics offscreen image of a view.

HIViewDrawCGImage (page 63)
Draws a Core Graphics image appropriately for a view.

Working with Grow Boxes

HIGrowBoxViewIsTransparent (page 28)
Determines whether a grow box view is transparent.

HIGrowBoxViewSetTransparent (page 28)
Makes a grow box view transparent or opaque.

Using Cocoa Views in Carbon Windows

`HICocoaViewCreate` (page 20)
Creates a Carbon view that serves as a wrapper for a Cocoa view.

`HICocoaViewSetView` (page 21)
Associates a Cocoa view with a HICocoaView wrapper view.

`HICocoaViewGetView` (page 21)
Returns the Cocoa view associated with an existing Carbon wrapper view.

Functions

HICocoaViewCreate

Creates a Carbon view that serves as a wrapper for a Cocoa view.

```
OSStatus HICocoaViewCreate (
    NSView *inNSView,
    OptionBits inOptions,
    HIViewRef *outHIView
);
```

Parameters

inNSView
A pointer to the Cocoa view you want to wrap. This function retains the Cocoa view you pass in; on output, you may safely release the view. If you want to create an empty Carbon wrapper view, you may pass `NULL`. An empty wrapper view does not draw or respond to user interaction; you can associate it with a Cocoa view at a later time using the function `HICocoaViewSetView` (page 21).

inOptions
Options for the new Carbon wrapper view. Currently this parameter must be 0.

outHIView
A pointer to a variable of type `HIViewRef` (page 110). On output, your variable contains a new Carbon view that serves as a wrapper for the Cocoa view specified in the `inNSView` parameter. You are responsible for releasing the wrapper view when you no longer need it. Note that if you embed the wrapper view in a Carbon window, the view (along with its associated Cocoa view) will be released automatically when the window is destroyed.

Return Value
An operating system result code. This function returns `paramErr` whenever the `inOptions` parameter is not 0 or the `outHIView` parameter is `NULL`.

Discussion
This function creates an HIView-based wrapper for a Cocoa view. You can embed the new wrapper view in a Carbon window and use standard HIView functions to manipulate the view. HICocoaView is supported only in compositing windows.

The following example shows how to use this function to create a wrapped Cocoa view that can be embedded in a Carbon window:

```
NSView *myCocoaView = [[SomeNSView alloc] init];
HIViewRef myHICocoaView;
```

```
HICocoaViewCreate (myCocoaView, 0, &myHICocoaView);
[myCocoaView release];
```

Availability
Available in Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`HICocoaView.h`

HICocoaViewGetView

Returns the Cocoa view associated with an existing Carbon wrapper view.

```
NSView * HICocoaViewGetView (
    HIViewRef inHIView
);
```

Parameters

`inHIView`

A wrapper view that has an associated Cocoa view.

Return Value
The Cocoa view associated with the specified wrapper view, or `NULL` if the wrapper view is empty or invalid. If you need to save the Cocoa view for later use, you should retain it.

Availability
Available in Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`HICocoaView.h`

HICocoaViewSetView

Associates a Cocoa view with a HICocoaView wrapper view.

```
OSStatus HICocoaViewSetView (
    HIViewRef inHIView,
    NSView *inNSView
);
```

Parameters

`inHIView`

An existing HICocoaView wrapper view.

`inNSView`

A pointer to a Cocoa view. This function retains the Cocoa view you pass in; on output, you may safely release this view. If the HICocoaView wrapper view specified in the `inHIView` parameter already wraps a Cocoa view, this function releases the wrapped view and replaces it with the Cocoa view you pass in.

Return Value
An operating system result code. This function returns `paramErr` if either parameter is `NULL` or invalid.

Discussion
Typically you'll use this function after you instantiate a nib-based Carbon window that contains an empty HICocoaView wrapper view. The empty wrapper view serves as a placeholder until you call this function to associate a Cocoa view with it. HICocoaView is supported only in compositing windows.

The following example shows how to use this function to associate a Cocoa view with an existing wrapper view:

```
NSView *myCocoaView = [[SomeNSView alloc] init];
HICocoaViewSetView (myHICocoaView, myCocoaView);
[myCocoaView release];
```

Availability
Available in Mac OS X v10.5 and later.
Not available to 64-bit applications.

Declared In
`HICocoaView.h`

HIComboBoxAppendTextItem

Appends a text item to the combo box disclosure list.

```
OSStatus HIComboBoxAppendTextItem (
   HIViewRef inComboBox,
   CFStringRef inText,
   CFIndex *outIndex
);
```

Parameters

inComboBox
The combo box having the disclosure list to which the text is to be appended.

inText
The text item to be appended to the combo box disclosure list.

outIndex
On exit, the index of the new item. Can be `NULL` if you don't want this information.

Return Value
An operating system status code.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIComboBox.h`

HIComboBoxChangeAttributes

Changes the attributes of a combo box.

```
OSStatus HIComboBoxChangeAttributes (
   HIViewRef inComboBox,
   OptionBits inAttributesToSet,
   OptionBits inAttributesToClear
);
```

Parameters

inComboBox

The combo box whose attributes you want to change.

inAttributesToSet

The attributes to set. For possible values, see "Combo Box Attributes" (page 113).

inAttributesToClear

The attributes to clear. For possible values, see "Combo Box Attributes" (page 113).

Return Value

An operating system status code.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIComboBox.h`

HIComboBoxCopyTextItemAtIndex

Copy a text item from a combo box disclosure list

```
OSStatus HIComboBoxCopyTextItemAtIndex (
   HIViewRef inComboBox,
   CFIndex inIndex,
   CFStringRef *outString
);
```

Parameters

inComboBox

The combo box that contains the text item you want to copy.

inIndex

The index of the text item. This function returns `paramErr` if the index is out of the bounds of the combo box disclosure list.

outString

A copy of the string at the specified index. You are responsible for releasing the string.

Return Value

An operating system status code.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIComboBox.h`

HIComboBoxCreate

Creates a combo box control.

```
OSStatus HIComboBoxCreate (
   const HIRect *boundsRect,
   CFStringRef text,
   const ControlFontStyleRec *style,
   CFArrayRef list,
   OptionBits inAttributes,
   HIViewRef *outComboBox
);
```

Parameters

boundsRect

The bounding box of the control.

text

The default text in the editable portion of the control. Can be `NULL`.

style

The font style of the both editable text and the text in the disclosure list. Can be `NULL`.

list

The default values available in the disclosure list. Can be `NULL`.

inAttributes

The default attributes of the combo box. For possible values, see "Combo Box Attributes" (page 113).

outComboBox

On exit, a pointer to a reference for the new control.

Discussion

The combo box can be used in compositing mode, as well as traditional Control Manager mode. When created, this view is invisible. To see the view, you must show the view by calling `HIViewSetVisible` (page 103).

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIComboBox.h`

HIComboBoxGetAttributes

Gets the attributes of a combo box.

```
OSStatus HIComboBoxGetAttributes (
   HIViewRef inComboBox,
   OptionBits *outAttributes
);
```

Parameters

inComboBox

The combo box whose attributes you want to obtain.

outAttributes
The attributes of the combo box. For possible values, see "Combo Box Attributes" (page 113).

Return Value
An operating system status code.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIComboBox.h

HIComboBoxGetItemCount

Gets the number of items in the combo box disclosure list.

```
ItemCount HIComboBoxGetItemCount (
   HIViewRef inComboBox
);
```

Parameters

inComboBox
The combo box.

Return Value
The number of items in the combo box disclosure list.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIComboBox.h

HIComboBoxInsertTextItemAtIndex

Inserts a CFString in a combo box disclosure list.

```
OSStatus HIComboBoxInsertTextItemAtIndex (
   HIViewRef inComboBox,
   CFIndex inIndex,
   CFStringRef inText
);
```

Parameters

inComboBox
The combo box having the disclosure list in which the text is to be inserted.

inIndex
The index at which the text should be inserted. If the index does not fall within the number of items in the combo box list, the text is appended to the end of the list.

inText
The text item to be inserted in the combo box disclosure list.

Return Value
An operating system status code.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIComboBox.h`

HIComboBoxIsListVisible

Determines whether a combo box disclosure list is visible.

```
Boolean HIComboBoxIsListVisible (
   HIViewRef inComboBox
);
```

Parameters

inComboBox
The Combo box whose disclosure list visibility is to be queried.

Return Value
A Boolean whose value is `true` if the combo box disclosure list is visible; otherwise, `false` to indicate that the combo box disclosure list is hidden.

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

Declared In
`HIComboBox.h`

HIComboBoxRemoveItemAtIndex

Removes an item from a combo box disclosure list.

```
OSStatus HIComboBoxRemoveItemAtIndex (
   HIViewRef inComboBox,
   CFIndex inIndex
);
```

Parameters

inComboBox
The combo box having the disclosure list that from which you want to remove an item.

inIndex
The index of the item to remove.

Return Value
An operating system status code.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIComboBox.h`

HIComboBoxSetListVisible

Hides or shows a combo box disclosure list.

```
OSStatus HIComboBoxSetListVisible (
   HIViewRef inComboBox,
   Boolean inVisible
);
```

Parameters

inComboBox

The combo box.

inVisible

A Boolean whose value is `true` to make the combo box disclosure list visible or `false` to hide the combo box list.

Return Value
An operating system result code.

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

Declared In
`HIComboBox.h`

HICreateTransformedCGImage

Creates a new Core Graphics image with the standard selected or disabled appearance.

```
OSStatus HICreateTransformedCGImage (
   CGImageRef inImage,
   OptionBits inTransform,
   CGImageRef *outImage
);
```

Parameters

inImage

The original image.

inTransform

The transformation to apply to the image. For possible values, see "Transformation Constants" (page 138).

outImage

A pointer to a value of type `CGImageRef` that, on return, contains the new image. You are responsible for releasing the image.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIGrowBoxViewIsTransparent

Determines whether a grow box view is transparent.

```
Boolean HIGrowBoxViewIsTransparent (
   HIViewRef inGrowBoxView
);
```

Parameters

inGrowBoxView
The grow box view reference to query.

Return Value
A Boolean value that is `true` if the grow box view is transparent; otherwise, `false` which indicates the grow box view is an opaque white square with grow lines.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIWindowViews.h`

HIGrowBoxViewSetTransparent

Makes a grow box view transparent or opaque.

```
OSStatus HIGrowBoxViewSetTransparent (
   HIViewRef inGrowBoxView,
   Boolean inTransparent
);
```

Parameters

inGrowBoxView
The grow box view reference.

inTransparent
Pass `true` to make the grow box view use its transparent look or `false` to make the grow box view use its opaque appearance.

Return Value
An operating system result code.

Discussion
This function sets a grow box view to be transparent, meaning the grow box lines are drawn over any content under it. When not transparent, the grow box is an opaque white square with the grow lines.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIWindowViews.h`

HIImageViewCopyImage

Obtains the image for an image view.

```
CGImageRef HIImageViewCopyImage (
   HIViewRef inView
);
```

Parameters

inView
The image view to query.

Return Value
A Core Graphics (Quartz) image reference, or `NULL` if there is no image set on the view or if the view ref is invalid.

Discussion
The image is retained, so you should release it when you are finished with it.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIImageViews.h`

HIImageViewCreate

Creates an image view.

```
OSStatus HIImageViewCreate (
   CGImageRef inImage,
   HIViewRef *outView
);
```

Parameters

inImage
An initial image, or `NULL`. You can use `SetControlData` to set the image later.

outControl
The new image view.

Return Value
An operating system result code.

Discussion
The view responds to the scrollable interface and can be used in a scrolling view. You can pass an image initially, or set one later.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIImageViews.h`

HIImageViewGetAlpha

Obtains the alpha value for a view.

```
CGFloat HIImageViewGetAlpha (
   HIViewRef inView
);
```

Parameters

inView
The image view to query.

Return Value
A floating point number representing the alpha from 0.0 through 1.0.

Discussion
An alpha of 1.0 means that the view is fully opaque, and alpha of 0.0 is means the view is fully transparent.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIImageViews.h`

HIImageViewGetScaleToFit

Determines whether an image will scale or clip to the view bounds.

```
Boolean HIImageViewGetScaleToFit (
   HIViewRef inView
);
```

Parameters

inView
The image view to query.

Return Value
A Boolean whose value is `true` if the image scales or `false` if the image clips.

Discussion
This function determines whether an image view will scale the image it displays to the view bounds or merely clip to the view bounds.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIImageViews.h

HIImageViewIsOpaque

Determines whether an image view is opaque.

```
Boolean HIImageViewIsOpaque (
   HIViewRef inView
);
```

Parameters

inView
The image view to query.

Return Value
A Boolean whose value is `true` if the image view is opaque; otherwise, `false`.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIImageViews.h

HIImageViewSetAlpha

Sets the alpha value for an image view.

```
OSStatus HIImageViewSetAlpha (
   HIViewRef inView,
   CGFloat inAlpha
);
```

Parameters

inView
The image view to affect.

inAlpha
The new alpha value.

Return Value
An operating system result code.

Discussion
Allows you to set the alpha for an image, making it more or less transparent. An alpha of 1.0 is fully opaque, and an alpha of 0.0 is fully transparent. The default alpha for an image is 1.0.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIImageViews.h

HIImageViewSetImage

Sets the image to display in an image view.

```
OSStatus HIImageViewSetImage (
   HIViewRef inView,
   CGImageRef inImage
);
```

Parameters

inView

The image view to affect.

inImage

The image to set.

Return Value
An operating system status code.

Discussion
The image passed in is retained by the view, so you may release the image after calling this function if you no longer need to reference it.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`HIImageViews.h`

HIImageViewSetOpaque

Sets the opacity of an image view.

```
OSStatus HIImageViewSetOpaque (
   HIViewRef inView,
   Boolean inOpaque
);
```

Parameters

inView

The image view to set.

inOpaque

A Boolean whose value is `true` to make the image view opaque or `false` to disable the opacity setting.

Return Value
An operating system result code.

Discussion
When opacity is enabled, the image view can make certain optimizations for compositing and scrolling. The alpha-related image view APIs are rendered useless when opacity is enabled. An image view, when created, is opaque by default. You must pass `false` to this function in order to change the alpha, etc. or if your image does not fill the full bounds of the view.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`HIImageViews.h`

HIImageViewSetScaleToFit

Specifies whether an image should scale or clip to the view's bounds.

```
OSStatus HIImageViewSetScaleToFit (
   HIViewRef inView,
   Boolean inScaleToFit
);
```

Parameters

inView
The image view.

inScaleToFit
A Boolean whose value is `true` to indicate that the image should be scaled to fit the view bounds or `false` to indicate that the image should clip to the view's bounds.

Return Value
An operating system status code.

Discussion
Normally, an image view clips to the view's bounds. Use this function to tell the image view to size the image to fit into the view's bounds.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIImageViews.h`

HIMenuGetContentView

Obtains an `HIViewRef` that can be used to draw menu content for a menu.

```
OSStatus HIMenuGetContentView (
   MenuRef inMenu,
   ThemeMenuType inMenuType,
   HIViewRef *outView
);
```

Parameters

inMenu

The menu for which an `HIViewRef` is to be obtained.

inMenuType

The type of menu for which the menu content view is to be returned. The same `MenuRef` may have multiple content views, depending on the menu type being displayed.

outView

A pointer to a value of type `HIViewRef` that, on return, represents the view, or `NULL` if the menu does not use an HIView to draw its content. The caller should not release this view.

Return Value

An operating system result code. If the menu uses an MDEF instead of a view to draw its content, this function sets `outView` to `NULL` and returns `noErr`.

Discussion

If the content view has not yet been created, the Menu Manager will create the content view using the view class ID and initialization event associated with the menu. Note that the menu content view is not the same as the window content view; the menu content view is embedded inside the window content view.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`Menus.h`

HIMenuViewGetMenu

Returns the `MenuRef` associated with a view that is a subclass of HIMenuView.

```
MenuRef HIMenuViewGetMenu (
   HIViewRef inView
);
```

Parameters

inView

The view whose menu is to be returned.

Return Value

The `MenuRef` associated with the specified view, or `NULL` if a view is passed that is not a subclass of HIMenuView.

Discussion

An HIMenuView subclass might use call this function to determine the menu it should draw.

Special Considerations

Prior to Mac OS X v10.5, this function returns `NULL` if passed an instance of the standard menu view. In Mac OS X v10.5 and later, this function correctly returns the owning menu of an instance of the standard menu view.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HIMenuView.h`

HIScrollViewCanNavigate

Determines whether it is possible to navigate in a scroll view.

```
Boolean HIScrollViewCanNavigate (
   HIViewRef inView,
   HIScrollViewAction inAction
);
```

Parameters

inView
The view to query.

inAction
The navigation action to test. For possible values, see "Scroll View Action Constants" (page 128).

Return Value
A Boolean whose value is `true` if the navigation specified by `inAction` is possible; otherwise, `false`.

Discussion
Use this function to determine whether it is possible to perform a particular navigation within a scroll view. For example, if a scroll view is already at the top of the scrollable content, it is not possible to navigate upward, so the home and page up actions would not be possible. You might use this function to help you update the state of menu items.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HIScrollView.h`

HIScrollViewCreate

Creates a scroll view.

```
OSStatus HIScrollViewCreate (
   OptionBits inOptions,
   HIViewRef *outView
);
```

Parameters

inOptions

Options for our scroll view. You must specify either a horizontal or a vertical scroll bar. If neither is passed, an error is returned. For possible values, see "Scroll View Constants" (page 127).

outView

The new scroll view.

Return Value
An operating system result code.

Discussion
This view has three parts. It can have a horizontal scroll bar, a vertical scroll bar, and a view to be scrolled that must be added by calling `HIViewAddSubview` (page 54). The added scroll view integrates itself automatically and appropriately.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
`HIScrollView.h`

HIScrollViewGetScrollBarAutoHide

Obtains current setting of a scroll view's scroll bar auto-hide setting.

```
Boolean HIScrollViewGetScrollBarAutoHide (
   HIViewRef inView
);
```

Parameters

inView

The view to query.

Return Value
A Boolean whose value is `true` if the auto-hide setting is enabled; otherwise, `false`.

Discussion
When the auto-hide setting is enabled, a scroll view's scroll bars are hidden when the entire scrollable view can be fully displayed in the scroll view's bounds. This is similar to the behavior of the Preview application.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIScrollView.h`

HIScrollViewNavigate

Changes the portion of a view's target.

```
OSStatus HIScrollViewNavigate (
   HIViewRef inView,
   HIScrollViewAction inAction
);
```

Parameters

inView

The view to scroll.

inAction

The action to take.

Return Value
An operating system result code.

Discussion
Use this function to programmatically change the portion of a scroll view's target. For example, you can call this function to move to the beginning or end of a document. You can also page up, down, left and right. In general, you should not call this function from embedded content, that is, the scrollable view inside the scroll view. Instead, for those cases, you should position yourself appropriately and tell the scroll view you changed via the `kEventScrollableInfoChanged` Carbon event. This function is merely a programmatic way to scroll as one would by hand using the scroll bars.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HIScrollView.h`

HIScrollViewSetScrollBarAutoHide

Sets a scroll view's auto-hide setting.

```
OSStatus HIScrollViewSetScrollBarAutoHide (
   HIViewRef inView,
   Boolean inAutoHide
);
```

Parameters

inView

The view to affect.

inAutoHide

A Boolean whose value is `true` to enable auto-hide and `false` to disable auto-hide.

Return Value
An operating system result code.

Discussion
When the auto-hide setting is enabled, a scroll view's scroll bars are hidden when the entire scrollable view can be fully displayed in the scroll view's bounds. This is similar to the behavior of the Preview application.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
`HIScrollView.h`

HISearchFieldChangeAttributes

Sets the attributes of a search field.

```
OSStatus HISearchFieldChangeAttributes (
   HIViewRef inSearchField,
   OptionBits inAttributesToSet,
   OptionBits inAttributesToClear
);
```

Parameters

inSearchField
The search field whose attributes are to be changed.

inAttributesToSet
The attributes to set. For possible values, see "Search Field Attribute Constants" (page 132).

inAttributesToClear
The attributes to clear. For possible values, see "Search Field Attribute Constants" (page 132).

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISearchField.h`

HISearchFieldCopyDescriptiveText

Obtains the description associated with a search field.

```
OSStatus HISearchFieldCopyDescriptiveText (
   HIViewRef inSearchField,
   CFStringRef *outDescription
);
```

Parameters

inSearchField
The search field whose descriptive text is to be obtained.

outDescription

A pointer to a value of type `CFStringRef` that, on return, represents the description that is associated with the search field specified by `inSearchField`. This parameter cannot be `NULL`. If no description is associated with the search field, on return, `outDescription` is set to `NULL`. If there is a description, a `CFStringRef` is created for you; you are responsible for releasing the `CFStringRef` when you no longer need it.

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`HISearchField.h`

HISearchFieldCreate

Creates a Search field control.

```
OSStatus HISearchFieldCreate (
   const HIRect *inBounds,
   OptionBits inAttributes,
   MenuRef inSearchMenu,
   CFStringRef inDescriptiveText,
   HIViewRef *outRef
);
```

Parameters

inBounds

The initial bounds of the view, or `NULL`. If this parameter is `NULL`, the view is created with empty bounds (0, 0, 0, 0).

inAttributes

The initial attributes of the search field. Indicates whether the field should contain the Cancel icon. For possible values, see "Search Field Attribute Constants" (page 132).

inSearchMenu

The menu to be associated with this search field, or `NULL`. If `inSearchMenu` is non-`NULL`, the menu will be retained by the search field and the Search icon will be enabled in the left side of the text field. If this parameter is `NULL`, the view will not display the Search icon in the left portion of the text field. You are expected to install handlers on this menu to handle the visual appearance of the menu (for example, to draw check marks or enable items when the menu receives the `kEventMenuEnableItems` Carbon event), and to keep track of what action should be performed by associating HICommands with each menu item and installing a handler for the `kEventClassCommand`/`kEventCommandProcess` Carbon event.

inDescriptiveText

The text to be displayed in the text field when the field does not have focus and contains no user-entered text, or `NULL`. This text should indicate the search criteria. For example, you may want to identify to the user that the search will cover the "Subject" or "Contents" of a selected range of items. If `inDescriptiveText` is non-`NULL`, the text will be retained by the search field.

outRef

On return, a reference for the new view.

Return Value
An operating system result code.

Discussion
This view is designed to be used by applications that provide searching functionality. Visually, it is a standard text field optionally adorned with a Search icon on the left and a Cancel image on the right. The new control is initially invisible.

When the user accepts the text by pressing the Return or Enter key, a Carbon event of `kEventClassTextField` / `kEventTextAccepted` is sent to the control to indicate that the search should begin. This control also responds to all of the standard control tags used by the `EditUnicodeText` control.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISearchField.h`

HISearchFieldGetAttributes

Obtains the attributes for a search field.

```
OSStatus HISearchFieldGetAttributes (
   HIViewRef inSearchField,
   OptionBits *outAttributes
);
```

Parameters

`inSearchField`
The search field whose attributes are to be obtained.

`outAttributes`
A pointer to a value of type `OptionBits` that, on return, contains the attributes of the Search field specified by `inSearchField`. This parameter cannot be `NULL`. For possible values, see "Search Field Attribute Constants" (page 132).

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISearchField.h`

HISearchFieldGetSearchMenu

Obtains the search menu associated with a search field.

```
OSStatus HISearchFieldGetSearchMenu (
   HIViewRef inSearchField,
   MenuRef *outSearchMenu
);
```

Parameters

inSearchField

The search field for which you want to obtain the search menu.

outSearchMenu

A pointer to a value of type `MenuRef` that, on return, represents the menu associated with the search field. The menu is *not* retained on output, and this parameter cannot be `NULL`.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In
`HISearchField.h`

HISearchFieldSetDescriptiveText

Sets the description of the search action for a search field.

```
OSStatus HISearchFieldSetDescriptiveText (
   HIViewRef inSearchField,
   CFStringRef inDescription
);
```

Parameters

inSearchField

The search field whose descriptive text is to be set.

inDescription

The new description for the search field. If the search field already has a description, it will be released. If this parameter is non-`NULL`, it will be retained by the search field. If this parameter is `NULL`, upon return, no description is associated with the search field.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In
`HISearchField.h`

HISearchFieldSetSearchMenu

Sets the search menu associated with a search field view.

```
OSStatus HISearchFieldSetSearchMenu (
   HIViewRef inSearchField,
   MenuRef inSearchMenu
);
```

Parameters

inSearchField

The Search field with which to associate the search menu.

inSearchMenu

The menu to associate with the Search field, or `NULL`. If a menu is already associated with the Search field, that menu is released. If `inSearchMenu` is non-`NULL`, it will be retained by the Search field and the Search icon will be enabled in the left side of the text field. You are expected to install handlers on this menu to handle the visual appearance of the menu (for example, to draw check marks or enable items when the menu receives the `kEventMenuEnableItems` Carbon event). You are also expected to keep track of the actions that should be performed by associating HICommands with each menu item and installing a handler for the `kEventClassCommand`/`kEventCommandProcess` Carbon event. If `inSearchMenu` is `NULL`, the Search icon is removed from the left side of the text field and no menu is associated with the Search field.

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`HISearchField.h`

HISegmentedViewChangeSegmentAttributes

Changes the attributes of an individual segment of a segmented view.

```
OSStatus HISegmentedViewChangeSegmentAttributes (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   OptionBits inAttributesToSet,
   OptionBits inAttributesToClear
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose attributes you want to change.

inSegmentIndexOneBased

The one-based index of the segment whose attributes you want to change. This must be a non-zero value that is no higher than the segmented view's current segment count.

inAttributesToSet

The attribute bits you want to set for the segment. For possible values, see "Segment Attribute Constants" (page 133).

inAttributesToClear

The attribute bits you want to clear for the segment.

Return Value
An operating system result code.

Discussion
By default, a segment has no attributes.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewCopySegmentImage

Copies the image drawn by a segment.

```
OSStatus HISegmentedViewCopySegmentImage (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   HIViewImageContentInfo *ioImage
);
```

Parameters

inSegmentedView
The segmented view that owns the segment whose image you want to copy.

inSegmentIndexOneBased
The one-based index of the segment whose image you want to set. This must be a non-zero value that is no higher than the segmented view's current segment count.

ioImage
A pointer to a `HIViewImageContentInfo` structure whose `contentType` field specifies the type of image you want to copy. If the segment uses the type of image you specified, on return, the appropriate field of the union contains a copy of the image. If the segment index is an illegal value, the result is undefined. You are responsible for releasing the image.

Return Value
An operating system result code.

Discussion
Call `HISegmentedViewGetSegmentImageContentType` (page 48) to get the image type.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewCopySegmentLabel

Obtains a copy of the label string associated with a segment.

```
OSStatus HISegmentedViewCopySegmentLabel (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   CFStringRef *outLabel
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose label you want to copy.

inSegmentIndexOneBased

The one-based index of the segment whose label is to be copied. This must be a non-zero value that is no higher than the segmented view's current segment count.

outLabel

A pointer to a `CFStringRef` that represents the copy of the label associated with the segment. You are responsible for releasing the the string containing the copy of the label. If no label is associated with the specified segment, `outLabel` is set to `NULL`.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewCreate

Creates a segmented view.

```
OSStatus HISegmentedViewCreate (
   const HIRect *inBounds,
   HIViewRef *outRef
);
```

Parameters

inBounds

The bounds of the view to be created, or `NULL`. If `NULL`, the view is created with `CGRectZero` bounds.

outRef

A valid pointer to an `HIViewRef` that, on return, represents the newly created view.

Return Value
An operating system result code.

Discussion
You can use a segmented view to implement the icon/column/list view switcher as seen in the Finder. After creating a segmented view, set the number of segments by calling `HISegmentedViewSetSegmentCount` (page 51). Each segment can be configured independently by calling other HISegmentedView APIs. Changing the number of segments and configuring each segment changes the appearance of the segmented view. After configuring the view, you may want to call `HIViewGetOptimalBounds` (page 73) on the view and resize it so the content fits optimally.

The value of the whole segmented view corresponds to the index of the currently selected segment with the radio behavior. If you set the value of the whole segmented view to *n* by calling `HIViewSetValue` (page 102), the value of each radio-behavior segment is set to zero except for the segment at index *n*. If segment *n* also has radio behavior, its value will be set to one. When a radio-behavior segment is clicked, the value of the whole segmented view is set to the segment's index.

Segmented views work in both compositing and non-compositing modes.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewGetSegmentAttributes

Returns the attributes of an individual segment of a segmented view.

```
OptionBits HISegmentedViewGetSegmentAttributes (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased
);
```

Parameters

inSegmentedView
The segmented view that owns the segment to query.

inSegmentIndexOneBased
The one-based index of the segment whose attributes you want to query. This must be a non-zero value that is no higher than the segmented view's current segment count.

Return Value
The attribute bits that are set for the specified segment. For possible values, see "Segment Attribute Constants" (page 133). If the segment index is an illegal value, the result is undefined.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewGetSegmentBehavior

Obtains the behavior of an individual segment of a segmented view.

```
HISegmentBehavior HISegmentedViewGetSegmentBehavior (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose behavior is to be obtained.

inSegmentIndexOneBased

The one-based index of the segment whose behavior you want to obtain. This must be a non-zero value that is no higher than the segmented view's current segment count.

Return Value

A value of type `HISegmentBehavior` describing the behavior of the given segment. For possible values, see "Segment Behavior Constants" (page 134). If the segment index is an illegal value, the result is undefined.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`HISegmentedView.h`

HISegmentedViewGetSegmentCommand

Obtains the command ID associated with a segment.

```
UInt32 HISegmentedViewGetSegmentCommand (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased
);
```

Parameters

inSegmentedView

The segmented view that owns the segment for which the command ID is to be obtained.

inSegmentIndexOneBased

The one-based index of the segment for which the command ID is to be obtained. This must be a non-zero value that is no higher than the segmented view's current segment count.

Return Value

The command ID associated with the specified segment. If the segment index is an illegal value, the result is undefined.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`HISegmentedView.h`

HISegmentedViewGetSegmentContentWidth

Obtains the content width of a segment.

```
CGFloat HISegmentedViewGetSegmentContentWidth (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   Boolean *outAutoCalculated
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose content width is to be obtained.

inSegmentIndexOneBased

The one-based index of the segment. This must be a non-zero value that is no higher than the segmented view's current segment count.

outAutoCalculateWidth

A pointer to a Boolean whose value, on return, is `true` if the segment calculates its own width automatically; otherwise, `false`. Pass `NULL` if you don't want this information.

Return Value
The width of the content for the given segment. If the segment index is an illegal value, the result is undefined.

Discussion
The content width is the horizontal area taken up by a segment's label (if any) and image (if any).

Availability
Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewGetSegmentCount

Obtains the number of segments for a segmented view.

```
UInt32 HISegmentedViewGetSegmentCount (
   HIViewRef inSegmentedView
);
```

Parameters

inSegmentedView

The segmented view for which the number of segments is to be obtained.

Return Value
A `UInt32` whose value is the number of segments in the segmented view specified by `inSegmentedView`.

Availability
Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewGetSegmentImageContentType

Obtains the type of image content drawn by a segment.

```
HIViewImageContentType HISegmentedViewGetSegmentImageContentType (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose image content type you want to obtain.

inSegmentIndexOneBased

The one-based index of the segment whose image you want to set. This must be a non-zero value that is no higher than the segmented view's current segment count.

Return Value

The image content type of the image drawn by the specified segment. If the segment index is an illegal value, the result is undefined.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`HISegmentedView.h`

HISegmentedViewGetSegmentValue

Returns the value of an individual segment of a segmented view.

```
SInt32 HISegmentedViewGetSegmentValue (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased
);
```

Parameters

inSegmentedView

The segmented view that owns the segment to query.

inSegmentIndexOneBased

The one-based index of the segment whose value you want to obtain. This must be a non-zero value that is no higher than the segmented view's current segment count.

Return Value

An `SInt32` containing the value of the specified segment. Zero means that the segment is unpressed or unselected, and one means the segment is pressed or selected.

Discussion

Getting a segment value is only meaningful for segments with the sticky, toggles, or radio behaviors. The value of segments that have the momentary behavior is undefined.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In
HISegmentedView.h

HISegmentedViewIsSegmentEnabled

Determines whether an individual segment of a segmented view is enabled.

```
Boolean HISegmentedViewIsSegmentEnabled (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased
);
```

Parameters

inSegmentedView

The segmented view that owns the segment to query.

inSegmentIndexOneBased

The one-based index of the segment to query. This must be a non-zero value that is no higher than the segmented view's current segment count.

Return Value
A Boolean whose value is `true` if the segment is enabled or `false` if the segment is not enabled.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
HISegmentedView.h

HISegmentedViewSetSegmentBehavior

Changes the behavior of an individual segment of a segmented view.

```
OSStatus HISegmentedViewSetSegmentBehavior (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   HISegmentBehavior inBehavior
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose behavior is to be set.

inSegmentIndexOneBased

The one-based index of the segment whose behavior you want to set. This must be a non-zero value that is no higher than the segmented view's current segment count.

inBehavior

The behavior you want the segment to have. For possible values, see "Segment Behavior Constants" (page 134).

Return Value
An operating system result code.

Discussion
By default, a segment has the `kHISegmentBehaviorMomentary` behavior.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentCommand

Sets the command ID for a segment.

```
OSStatus HISegmentedViewSetSegmentCommand (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   UInt32 inCommand
);
```

Parameters

`inSegmentedView`
The segmented view that owns the segment for which the command ID is to be set.

`inSegmentIndexOneBased`
The one-based index of the segment for which the command ID is to be set. This must be a non-zero value that is no higher than the segmented view's current segment count.

`inCommand`
The command ID you want to associate with the segment. When the command ID is `0` for a segment, the `kEventCommandProcess` event is not sent when the segment is clicked.

Return Value
An operating system result code.

Discussion
When any non-zero command ID is set, the segmented view sends an HICommand whenever the segment is clicked. By default, the command is sent to the segmented view and up the containment hierarchy. If you want the command to start at the user focus instead, set the `kHISegmentCmdToUserFocus` attribute for the segment.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentContentWidth

Specifies how the content width of segment is to be calculated.

```
OSStatus HISegmentedViewSetSegmentContentWidth (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   Boolean inAutoCalculateWidth,
   CGFloat inWidth
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose method of calculating content width you want to specify.

inSegmentIndexOneBased

The one-based index of the segment. This must be a non-zero value that is no higher than the segmented view's current segment count.

inAutoCalculateWidth

A Boolean whose value is `true` if you want the segment to calculate its own width automatically (in which case, the `inWidth` parameter is ignored), or `false` if you want the value of the `inWidth` parameter to be associated with the segment.

inWidth

The width in pixels.

Return Value
An operating system result code.

Discussion
The content width is the horizontal area taken up by a segment's label (if any) and image (if any).

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentCount

Sets the number of segments for a segmented view.

```
OSStatus HISegmentedViewSetSegmentCount (
   HIViewRef inSegmentedView,
   UInt32 inSegmentCount
);
```

Parameters

inSegmentedView

The segmented menu for which the number of segments is to be set.

inSegmentCount

A positive integer specifying the number of segments the view is to have.

Return Value
An operating system result code.

Discussion
The content for any previous segments beyond the new count is released. All new segments beyond the previous count are initialized with basic settings: Momentary, no attributes, zero value, enabled, no command, no label, no content, and auto-calculated content width. You should configure any new segments to match your needs.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentEnabled

Enables or disables an individual segment of a segmented view.

```
OSStatus HISegmentedViewSetSegmentEnabled (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   Boolean inEnabled
);
```

Parameters

inSegmentedView
The segmented view that owns the segment that is to be enabled or disabled.

inSegmentIndexOneBased
The one-based index of the segment to enable or disable. This must be a non-zero value that is no higher than the segmented view's current segment count.

inEnabled
A Boolean whose value is `true` to enable the segment or `false` to disable the segment.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentImage

Sets or clears the image associated with a segment.

```
OSStatus HISegmentedViewSetSegmentImage (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   const HIViewImageContentInfo *inImage
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose image you want to set.

inSegmentIndexOneBased

The one-based index of the segment whose image you want to set. This must be a non-zero value that is no higher than the segmented view's current segment count.

inImage

A pointer to an `HIViewImageContentInfo` structure with the image information for the specified segment. Segments support three types of image content: `kControlNoContent`, `kControlContentIconRef`, and `kControlContentCGImageRef`.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentLabel

Sets the label string for a segment.

```
OSStatus HISegmentedViewSetSegmentLabel (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   CFStringRef inLabel
);
```

Parameters

inSegmentedView

The segmented view that owns the segment whose label you want to sets.

inSegmentIndexOneBased

The one-based index of the segment whose label is to be set. This must be a non-zero value that is no higher than the segmented view's current segment count.

inLabel

A `CFStringRef` with the text of the label. The segmented view will copy the string you passed in. To eliminate the label from the segment, pass `NULL` or an empty `CFStringRef`.

Return Value
An operating system result code.

Discussion
By default, a segment has no label string.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HISegmentedViewSetSegmentValue

Changes the value of an individual segment of a segmented view.

```
OSStatus HISegmentedViewSetSegmentValue (
   HIViewRef inSegmentedView,
   UInt32 inSegmentIndexOneBased,
   SInt32 inValue
);
```

Parameters

inSegmentedView
The segmented view that owns the segment to query.

inSegmentIndexOneBased
The one-based index of the segment whose value you want to change. This must be a non-zero value that is no higher than the segmented view's current segment count.

inValue
The value you want to set. Zero means that the segment is unpressed or unselected, and one means the segment is pressed or selected. Setting any other value will result in an undefined behavior.

Return Value
An operating system result code.

Discussion
Setting a segment value is only meaningful for segments with the sticky, toggles, or radio behaviors. Setting the value of segments that have the momentary behavior to something other than zero results in a behavior that is undefined.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HISegmentedView.h`

HIViewAddSubview

Adds a subview to the given parent view.

```
OSStatus HIViewAddSubview (
   HIViewRef inParent,
   HIViewRef inNewChild
);
```

Parameters

inParent

The view that will receive the new subview.

inNewChild

The subview being added.

Return Value

An operating system result code. The result code `errNeedsCompositedWindow` is returned if you try to embed into the content view in a non-compositing window; you can only embed into the content view in a compositing window.

Discussion

The new subview is added to the front of the list of subviews (that is, it is made topmost). The subview being added is not retained by the new parent view. Do not release the view after adding it, or it will cease to exist. All views in a window are released automatically when the window is destroyed.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Related Sample Code

HID Calibrator

Declared In

`HIView.h`

HIViewAdvanceFocus

Advances the keyboard focus to the next most appropriate view.

```
OSStatus HIViewAdvanceFocus (
   HIViewRef inRootForFocus,
   EventModifiers inModifiers
);
```

Parameters

inRootForFocus

The subtree to manipulate. The focus does not leave `inRootToFocus`. Typically, you would pass the content of the window or the root. If focused on the toolbar, for example, the focus is limited to the toolbar only. In this case, for example, the Toolbox passes the toolbar view in as the focus root.

inModifiers

The keyboard event modifiers that caused `HIViewAdvanceFocus` (page 55) to be called. These modifiers are used to determine the focus direction as well as other alternate focusing behaviors.

Return Value

An operating system result code.

Discussion
Unless overridden in some fashion (either by overriding certain Carbon events or by calling the `HIViewSetNextFocus` (page 100), the Toolbox uses a spatially determinant method of focusing, attempting to focus left to right and top to bottom in a window, taking groups of controls into account.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewApplyLayout

Applies the current layout to the specified view.

```
OSStatus HIViewApplyLayout (
   HIViewRef inView
);
```

Parameters

inView
The view to which the layout is to be applied.

Return Value
An operating system result code.

Discussion
Side bindings have no effect, but positioning and scaling are applied, in that order.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewChangeAttributes

Changes the attributes of a view.

```
OSStatus HIViewChangeAttributes (
   HIViewRef inView,
   OptionBits inAttrsToSet,
   OptionBits inAttrsToClear
);
```

Parameters

inView
The view whose attributes you want to change.

inAttrsToSet
The attributes you want to change. For possible values, see "HIView Attributes" (page 121).

inAttrsToClear

The attributes you want to clear. For possible values, see "HIView Attributes" (page 121).

Return Value
An operating system result code.

Discussion
You can set and clear attributes simultaneously.

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

Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewChangeFeatures

Changes the features of a view.

```
OSStatus HIViewChangeFeatures (
   HIViewRef inView,
   HIViewFeatures inFeaturesToSet,
   HIViewFeatures inFeaturesToClear
);
```

Parameters

inView

The view whose features are to be changed.

inFeaturesToSet

The features that are to be set. For details, see "HIView Feature Constants" (page 122).

inFeaturesToClear

The features that are to be cleared. For details, see "HIView Feature Constants" (page 122).

Return Value
An operating system result code.

Discussion
The view itself typically controls its features. For example, the view might decide that under some situations it is opaque and in others it is transparent. In general entities outside of the view itself should not call this function. The only exception might be user-interface building tools that want to make sure a view always responds to clicks, for example, so it could override mouse tracking to drag items.

Availability
Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
`HIView.h`

HIViewChangeTrackingArea

Changes the shape of a tracking area.

```
OSStatus HIViewChangeTrackingArea (
   HIViewTrackingAreaRef inArea,
   HIShapeRef inShape
);
```

Parameters

inArea

The tracking area to change.

inShape

The shape to use. Pass `NULL` to use the entire structure region of the view.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewClick

Passes a mouse-down event to a view.

```
OSStatus HIViewClick (
   HIViewRef inView,
   EventRef inEvent
);
```

Parameters

inView

The view to handle the event.

inEvent

The mouse event to handle.

Return Value
An operating system result code.

Discussion
After a successful call to `HIViewGetViewForMouseEvent` for a mouse down event, you should call this function to have the view handle the click. In general we recommend using the Standard Window Handler instead of calling this function yourself.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewConvertPoint

Converts a point's coordinates from one view to another.

```
OSStatus HIViewConvertPoint (
   HIPoint *ioPoint,
   HIViewRef inSourceView,
   HIViewRef inDestView
);
```

Parameters

ioPoint

The point to convert.

inSourceView

The view whose coordinate system `ioPoint` is starting out in. You can pass `NULL` to indicate that `ioPoint` is a window-relative point.

inDestView

The view whose coordinate system `ioPoint` should end up in. You can pass `NULL` to indicate that `ioPoint` is a window-relative point.

Return Value

An operating system result code.

Discussion

Both views must have a common ancestor, that is, they must both be in the same window.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewConvertRect

Converts a rectangle from one view to another.

```
OSStatus HIViewConvertRect (
   HIRect *ioRect,
   HIViewRef inSourceView,
   HIViewRef inDestView
);
```

Parameters

ioRect

The rectangle to convert.

inSourceView

The view whose coordinate system `ioRect` is starting out in. You can pass `NULL` to indicate that `ioRect` is a window-relative rectangle.

inDestView

The view whose coordinate system `ioRect` should end up in. You can pass `NULL` to indicate that `ioRect` is a window-relative rectangle.

Return Value
An operating system result code.

Discussion
Both views must have a common ancestor, that is, they must both be in the same window.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewConvertRegion

Converts a region from one view to another.

```
OSStatus HIViewConvertRegion (
   RgnHandle ioRgn,
   HIViewRef inSourceView,
   HIViewRef inDestView
);
```

Parameters

ioRgn
The region to convert.

inSourceView
The view whose coordinate system `ioRgn` is starting out in. You can pass `NULL` to indicate that `ioRgn` is a window-relative region.

inDestView
The view whose coordinate system `ioRgn` should end up in. You can pass `NULL` to indicate that `ioRgn` is a window-relative region.

Return Value
An operating system result code.

Discussion
Both views must have a common ancestor, that is, they must both be in the same window.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewCopyShape

Copies the shape of a part of a view.

```
OSStatus HIViewCopyShape (
   HIViewRef inView,
   HIViewPartCode inPart,
   HIShapeRef *outShape
);
```

Parameters

inView

The view having a part whose shape is to be copied.

inPart

The part of the view whose shape is to be copied. For possible values, see "HIViewPartCode Constants" (page 126).

outShape

On exit, the newly created shape. You are responsible for releasing the copied shape.

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewCopyText

Copies the text of a view.

```
CFStringRef HIViewCopyText (
   HIViewRef inView
);
```

Parameters

inView

The view whose text is to be copied.

Return Value

A CFString containing a copy of the view's text. The caller is responsible for releasing the CFString.

Discussion

This function attempts to copy the text that is displayed when drawing the view and is generally successful on views that handle the `kControlEditTextCFStringTag` `GetControlData` tag. If this function can't copy that text, it copies the text in the view's title instead.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewCountSubviews

Returns the number of subviews embedded in a view.

```
CFIndex HIViewCountSubviews (
   HIViewRef inView
);
```

Parameters

inView

The view whose subviews are to be counted.

Return Value

The number of subviews for the specified view.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewCreateOffscreenImage

Creates a Core Graphics offscreen image of a view.

```
OSStatus HIViewCreateOffscreenImage (
   HIViewRef inView,
   OptionBits inOptions,
   HIRect *outFrame,
   CGImageRef *outImage
);
```

Parameters

inView

The view you want to create an image of.

inOptions

Options. Currently you must pass zero.

outFrame

The frame of the view within the resultant image. It is in the coordinate system of the image, where 0,0 is the top left corner of the image. This is so you can know exactly where the control is in the image when the control draws outside its bounds for things such as shadows.

outImage

The image of the view, including anything that would be drawn outside the view's frame.

Return Value

An operating system status code.

Discussion

This function creates an `CGImageRef` for the specified view. The view and any of its children are rendered in the resultant image.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In
HIView.h

HIViewDisposeTrackingArea

Disposes of an existing tracking area.

```
OSStatus HIViewDisposeTrackingArea (
   HIViewTrackingAreaRef inArea
);
```

Parameters

inArea
The tracking area that is to be disposed of.

Return Value
An operating system result code.

Discussion
All tracking areas attached to a view are automatically disposed of when the view is disposed of, so you don't need to dispose of a tracking area explicitly unless you want to remove it from a view before the view is disposed of.

After you call this function, the reference is invalid.

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

Declared In
HIView.h

HIViewDrawCGImage

Draws a Core Graphics image appropriately for a view.

```
OSStatus HIViewDrawCGImage (
   CGContextRef inContext,
   const HIRect *inBounds,
   CGImageRef inImage
);
```

Parameters

inContext
The context to draw in.

inBounds
The bounds to draw the image into.

inImage
The image to draw.

Return Value
An operating system status code.

Discussion
This function is similar to `CGContextDrawImage`, but it flips the context so that the image is drawn correctly. The origin of a view is at the top left corner, so you are really drawing upside-down. This call insulates you from that fact.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewFindByID

Obtains a view by its ID.

```
OSStatus HIViewFindByID (
   HIViewRef inStartView,
   HIViewID inID,
   HIViewRef *outView
);
```

Parameters

inStartView
The view to start searching at.

inID
The ID of the view you are looking for.

outControl
Receives the control if found.

Return Value
An operating system result code.

Discussion
Allows you to find a particular view by its ID. The `HIViewID` type used by this function is identical to the `ControlID` type used by the Control Manager.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
CarbonCocoa_PictureCursor
HID Calibrator
HID Config Save
QTCarbonShell
QTMetaData

Declared In
`HIView.h`

HIViewFlashDirtyArea

Flashes a window's dirty area.

```
OSStatus HIViewFlashDirtyArea (
   WindowRef inWindow
);
```

Parameters

inWindow

The window whose dirty area is to be flashed.

Return Value

An operating system result code.

Discussion

As a debugging aid, this function flashes the area for an entire window that will be redrawn at the next draw time.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIToolboxDebugging.h`

HIViewGetAttributes

Obtains the attributes for a view.

```
OSStatus HIViewGetAttributes (
   HIViewRef inView,
   OptionBits *outAttrs
);
```

Parameters

inView

The view to inspect.

outAttrs

The attributes of the view. For possible values, see "HIView Attributes" (page 121).

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetBounds

Obtains the local bounds of a view.

```
OSStatus HIViewGetBounds (
   HIViewRef inView,
   HIRect *outRect
);
```

Parameters

inView

The view whose local bounds are to be obtained.

outRect

The local bounds of the view.

Return Value
An operating system result code.

Discussion
The local bounds are the coordinate system that is completely view-relative. A view's top left coordinate starts out at 0, 0. Most operations use local coordinates. Note, however, that the frame is used to move a view, not local coordinates.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator
HID Config Save

Declared In
`HIView.h`

HIViewGetCommandID

Obtains the command ID of a view.

```
OSStatus HIViewGetCommandID (
   HIViewRef inView,
   UInt32 *outCommandID
);
```

Parameters

inView

The view whose command ID is to be obtained.

outID

A pointer to a value of type `UInt32` that, on return, contains the view's command ID.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewGetEventTarget

Returns the `EventTargetRef` for the specified view.

```
EventTargetRef HIViewGetEventTarget (
   HIViewRef inView
);
```

Parameters

inImage

The view for which the `EventTargetRef` should be returned.

Return Value

The `EventTargetRef`.

Discussion

Once you obtain this reference, you can install an event handler and send events to the target.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetFeatures

Obtains the features of the specified view.

```
OSStatus HIViewGetFeatures (
   HIViewRef inView,
   HIViewFeatures *outFeatures
);
```

Parameters

inView

The view to query.

outFeatures

A pointer to a value of the `HIViewFeatures` that, on return, contains the view's features. For more information, see "HIView Feature Constants" (page 122).

Return Value

An operating system result code.

Discussion

This function returns feature bits for the view but does not return older Control Manager features, such as `kControlSupportsDataAccess`.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetFirstSubview

Returns the first subview of a parent view.

```
HIViewRef HIViewGetFirstSubview (
   HIViewRef inView
);
```

Parameters

inView

The view whose subview you are fetching.

Return Value

An HIView reference, or `NULL` if this view has no subviews or is invalid.

Discussion

Returns the first subview of a container. The first subview is the topmost subview in z-order.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetFocusPart

Obtains the part in the specified view that currently has focus.

```
OSStatus HIViewGetFocusPart (
   HIViewRef inView,
   HIViewPartCode *outFocusPart
);
```

Parameters

inView

The view to inquire about.

outFocusPart

The part that currently has focus. For more information, see "HIViewPartCode Constants" (page 126).

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetFrame

Obtains the frame bounds of a view.

```
OSStatus HIViewGetFrame (
   HIViewRef inView,
   HIRect *outRect
);
```

Parameters

inView

The view whose frame you want to obtain.

outRect

The frame of the view.

Return Value
An operating system result code.

Discussion
The frame bounds is the bounds of a view relative to its parent's local coordinate system.

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

Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
`HIView.h`

HIViewGetID

Obtains the HIViewID of a view.

```
OSStatus HIViewGetID (
   HIViewRef inView,
   HIViewID *outID
);
```

Parameters

inView

The view whose validity is to be checked.

outID

A pointer to a value of type `HIViewID` that, on return, contains the view's `HIViewID`.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewGetIndexedSubview

Obtains the subview of a view by index.

```
OSStatus HIViewGetIndexedSubview (
   HIViewRef inView,
   CFIndex inSubviewIndex,
   HIViewRef *outSubview
);
```

Parameters

inView

The view whose indexed subview is being requested.

inSubviewIndex

The index of the requested subview.

outSubview

A pointer to an `HIViewRef` that, on return, represents the indexed subview.

Return Value

The number of subviews for the specified view.

Discussion

Instead of calling `HIViewGetIndexedSubview` repeatedly, it may be more efficient to iterate through the subviews of a view with calls to `HIViewGetFirstSubview` (page 68) and `HIViewGetNextView` (page 73).

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetKind

Obtains the signature and kind of a view.

```
OSStatus HIViewGetKind (
   HIViewRef inView,
   HIViewKind *outViewKind
);
```

Parameters

inView

The view whose signature and kind is to be obtained.

outViewKind

A pointer to a `HIViewKind` structure that, on return, contains the view's signature and kind. For details, see HIViewKind (page 110).

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In
HIView.h

HIViewGetLastSubview

Returns the last subview in a parent view.

```
HIViewRef HIViewGetLastSubview (
   HIViewRef inView
);
```

Parameters

inView
The view whose subview you are fetching.

Return Value
An HIView reference, or NULL if this view has no subviews or is invalid.

Discussion
Returns the last subview of a container. The last subview is the bottommost subview in z-order.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIView.h

HIViewGetLayoutInfo

Obtains the layout information of an view.

```
OSStatus HIViewGetLayoutInfo (
   HIViewRef inView,
   HILayoutInfo *outLayoutInfo
);
```

Parameters

inOptions
The view whose layout information is to be obtained.

outLayoutInfo
A pointer to an HILayoutInfo (page 106) into which to copy the view's layout information. If the version field of this structure is not valid, the call will fail.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
HIView.h

HIViewGetMaximum

Obtains a view's maximum value.

```
SInt32 HIViewGetMaximum (
   HIViewRef inView
);
```

Parameters

inView
The view whose maximum value is to be obtained.

Return Value
The maximum value of the specified view.

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

Declared In
HIView.h

HIViewGetMinimum

Obtains the minimum value of a view.

```
SInt32 HIViewGetMinimum (
   HIViewRef inView
);
```

Parameters

inView
The view whose minimum value is to be obtained.

Return Value
The minimum value of the specified view.

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

Declared In
HIView.h

HIViewGetNeedsDisplay

Determines whether a view needs to be redrawn.

```
Boolean HIViewGetNeedsDisplay (
   HIViewRef inView
);
```

Parameters

inView

The view to inspect.

Return Value

A Boolean whose value is `true` if the view passed in or any of its subviews require redrawing; otherwise, `false`.

Discussion

A view or subview requires redrawing if any part of the view or subview has been invalidated.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetNextView

Returns the view behind the specified view.

```
HIViewRef HIViewGetNextView (
   HIViewRef inView
);
```

Parameters

inView

The view to use as reference.

Return Value

An HIView reference, or `NULL` if this view has no view behind it or is invalid.

Discussion

Returns the view after the specified view, in z-order.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewGetOptimalBounds

Obtains the optimal size and text placement of a view.

```
OSStatus HIViewGetOptimalBounds (
   HIViewRef inView,
   HIRect *outBounds,
   CGFloat *outBaseLineOffset
);
```

Parameters

inView

The view whose optimal size and text placement are to be obtained.

outBounds

A pointer to a value of type `HIRect` that, on return, contains the view's optimal bounds. Pass `NULL` if you don't need this information.

outBaseLineOffset

A pointer to a value of type `float` that, on return, contains the view's optimal text placement. Pass `NULL` if you don't need this information.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewGetPartHit

Determines the part hit for a given point.

```
OSStatus HIViewGetPartHit (
   HIViewRef inView,
   const HIPoint *inPoint,
   HIViewPartCode *outPart
);
```

Parameters

inView

The view to test the part hit.

inPoint

The view-relative point to use.

outPart

The part hit by `inPoint`.

Return Value
An operating system result code.

Discussion
Given a view, and a view-relative point, this function returns the part code hit as determined by the view.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIView.h

HIViewGetPreviousView

Returns the view above the specified view.

```
HIViewRef HIViewGetPreviousView (
   HIViewRef inView
);
```

Parameters

inView
The view to use as reference.

Return Value
An HIView reference, or NULL if this view has no view in front of it or is invalid.

Discussion
Returns the view before the specified view, in z-order.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIView.h

HIViewGetRoot

Obtains the root view for a window.

```
HIViewRef HIViewGetRoot (
   WindowRef inWindow
);
```

Parameters

inWindow
The window to get the root for.

Return Value
The root view for the window, or NULL if an invalid window is passed.

Discussion
Note that the root view is not the same as the Control Manager root control.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
CarbonCocoa_PictureCursor
HID Calibrator
HID Config Save

QTCarbonShell
QTMetaData

Declared In
`HIView.h`

HIViewGetSizeConstraints

Returns the minimum and maximum size for a control.

```
OSStatus HIViewGetSizeConstraints (
   HIViewRef inView,
   HISize *outMinSize,
   HISize *outMaxSize
);
```

Parameters

inView
The view to inspect.

outMinSize
The minimum size the view can be.

outMaxSize
The maximum size the view can be.

Return Value
An operating system result code.

Discussion
These sizes can, for example, be used to help auto-position subviews. To get meaningful results, the control must respond to the `kEventControlGetSizeContraints` event.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewGetSubviewHit

Returns the child of the given view hit by the point passed in.

```
OSStatus HIViewGetSubviewHit (
   HIViewRef inView,
   const HIPoint *inPoint,
   Boolean inDeep,
   HIViewRef *outView
);
```

Parameters

inView
The view you wish to position.

inPoint

The mouse coordinate to use. This is passed in the local coordinate system of inView.

inDeep

Pass true to find the deepest child hit, false to go only one level deep (just check direct children of inView).

outView

The view hit by inPoint, or NULL if no subview was hit.

Return Value
An operating system result code.

Discussion
This function is more primitive than using `HIViewGetViewForMouseEvent` (page 78), and should be used only in non-event-handling cases.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewGetSuperview

Returns a view's parent view.

```
HIViewRef HIViewGetSuperview (
   HIViewRef inView
);
```

Parameters

inView

The view whose parent you are interested in getting.

Return Value
An HIView reference, or `NULL` if this view has no parent or is invalid.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewGetTrackingAreaID

Obtains the ID of a tracking area.

```
OSStatus HIViewGetTrackingAreaID (
   HIViewTrackingAreaRef inArea,
   HIViewTrackingAreaID *outID
);
```

Parameters

inArea

The tracking area whose ID is to be obtained.

outID

On return, the ID for the tracking area specified by `inArea`.

Return Value
An operating system result code.

Discussion
The tracking area ID that is obtained is the value that was specified when `HIViewNewTrackingArea` (page 85) was called to create the tracking area.

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

Declared In
`HIView.h`

HIViewGetValue

Obtains the value of a view.

```
SInt32 HIViewGetValue (
   HIViewRef inView
);
```

Parameters

inView

The view whose value is to be obtained.

Return Value
The view's value.

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

Declared In
`HIView.h`

HIViewGetViewForMouseEvent

Returns the appropriate view to handle a mouse event.

```
OSStatus HIViewGetViewForMouseEvent (
   HIViewRef inView,
   EventRef inEvent,
   HIViewRef *outView
);
```

Parameters

inView

The view to start from. You should pass the window's root view.

inEvent

The mouse event in question.

outView

The view that the mouse event should be sent to.

Return Value
An operating system result code.

Discussion
This function is a little higher-level than `HIViewGetSubviewHit` (page 76). This function finds the deepest view that should handle the mouse event. It also sends a Carbon event to each view asking it to return the appropriate subview, which allows parent views to catch clicks on their subviews. This function is the recommended function to use before processing mouse events. Using one of the more primitive functions may result in an undefined behavior.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewGetViewSize

Obtains the view size of a view.

```
SInt32 HIViewGetViewSize (
   HIViewRef inView
);
```

Parameters

inView

The view whose view size is to be obtained.

Return Value
The view size.

Discussion
The view size is the size of the content to which a view's display is proportioned. The view size is commonly used to set the proportional size of a scroll bar's thumb indicator.

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

Declared In
HIView.h

HIViewGetWindow

Obtains a reference to the window to which the specified view is bound.

```
WindowRef HIViewGetWindow (
   HIViewRef inView
);
```

Parameters

inView

The view to query.

Return Value
An operating system result code, or NULL if the view reference specified by inView is invalid or if the view is not bound to any window.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
HIView.h

HIViewIsActive

Determines whether a view is active.

```
Boolean HIViewIsActive (
   HIViewRef inView,
   Boolean *outIsLatentActive
);
```

Parameters

inView

The view that is to be queried.

outIsLatentActive

A pointer to a Boolean that, on return, is set to true if the view is latently active or false if the view is not latently active. Pass NULL if you don't need this information.

Return Value
A Boolean whose value indicates whether the view is active (true) or not (false).

Discussion
A view's active state is affected by the active state of its parents. If HIViewIsActive finds that any parent view is inactive, it returns false to indicate that the view specified by inView is considered to be inactive too. In addition, HIViewIsActive can optionally check to see if a view is latently active, even if the view's parents are inactive.

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

Declared In
HIView.h

HIViewIsCompositingEnabled

Determines whether compositing is enabled for a view.

```
Boolean HIViewIsCompositingEnabled (
   HIViewRef inView
);
```

Parameters

inView

The view that is to be queried.

Return Value
A Boolean whose value indicates whether compositing is enabled (`true`) or not (`false`).

Discussion
Checking a window's `kWindowCompositingAttribute` attribute is not sufficient for determining whether a view is in compositing or non-compositing mode because some of a window's views can be in either mode at the same time. Call this function to determine the current compositing mode of a view.

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

Declared In
HIView.h

HIViewIsDrawingEnabled

Determines if drawing is currently enabled for a view.

```
Boolean HIViewIsDrawingEnabled (
   HIViewRef inView
);
```

Parameters

inView

The view to get the drawing state for.

Return Value
A Boolean value indicating that drawing is on (`true`) or off (`false`).

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
HIView.h

HIViewIsEnabled

Determines whether a view is enabled.

```
Boolean HIViewIsEnabled (
   HIViewRef inView,
   Boolean *outIsLatentEnabled
);
```

Parameters

inView

The view to query.

outIsLatentEnabled

A pointer to a Boolean that, on return, is set to `true` if the view is latently enabled or `false` if the view is not latently enabled. Pass `NULL` if you don't need this information.

Return Value

A Boolean whose value indicates whether the view is enabled (`true`) or not (`false`).

Discussion

A view's enabled state is affected by the enabled state of its parents. If `HIViewIsEnabled` finds that any parent view is disabled, it returns `false` to indicate that the view specified by `inView` is considered to be disabled too. In addition, `HIViewIsEnabled` can optionally check to see if a view is latently enabled, even if its parents are disabled.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In

`HIView.h`

HIViewIsLatentlyVisible

Determines whether a view is latently visible.

```
Boolean HIViewIsLatentlyVisible (
   HIViewRef inView
);
```

Parameters

inView

The view whose latent visibility is to be queried.

Return Value

A Boolean value indicating whether the view is latently visible (`true`) or hidden (`false`).

Discussion

A view's visibility is affected by the visibility of its parents. If any parent view is invisible, the view specified by `inView` is considered to be invisible too. `HIViewIsLatentlyVisible` returns whether a view is latently visible, even if its parents are invisible.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In
HIView.h

HIViewIsLayoutActive

Determines whether layout handling is active or suspended.

```
Boolean HIViewIsLayoutActive (
   HIViewRef inView
);
```

Parameters

inView
The view for which the status of layout handling is to be determined.

Return Value
A Boolean whose value is `true` if the view would respond to any linked relative's changes; otherwise `false`.

Discussion
A view's layout active state is also affected by the layout active state of its parents. If a parent view has an inactive layout, this view is also considered to have an inactive layout. To determine the latent active state of a view, see `HIViewIsLayoutLatentlyActive` (page 83).

Note that this function does not determine whether the view's layout is valid.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
HIView.h

HIViewIsLayoutLatentlyActive

Determines whether layout handling is latently active or suspended.

```
Boolean HIViewIsLayoutLatentlyActive (
   HIViewRef inView
);
```

Parameters

inView
The view for which the status of layout handling is to be determined.

Return Value
A Boolean whose value is `true` if the view would latently respond to any linked relative's changes; otherwise `false`.

Discussion
This function determines whether a view's layout is latently active, even if one of its parent's layouts is not active.

Availability
Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewIsValid

Determines whether the specified view is known to the HIToolbox.

```
Boolean HIViewIsValid (
   HIViewRef inView
);
```

Parameters

inView
The view to check.

Return Value
A Boolean whose value is `true` if the view is known to the HIToolbox; otherwise, `false`.

Discussion
This function does *not* check the data in the view for validity.

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

Declared In
`HIView.h`

HIViewIsVisible

Determines whether a view is visible.

```
Boolean HIViewIsVisible (
   HIViewRef inView
);
```

Parameters

inView
The view whose visibility you want to determine.

Return Value
A Boolean value indicating whether the view is visible (`true`) or hidden (`false`).

Discussion
`HIViewIsVisible` returns the effective visibility of a view, which is determined both by the view's own visibility and the visibility of its parent views. If a parent view is invisible, this view is considered to be invisible too.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewMoveBy

Move a view by the specified distance relative to its current location.

```
OSStatus HIViewMoveBy (
   HIViewRef inView,
   CGFloat inDX,
   CGFloat inDY
);
```

Parameters

inView

The view you want to move.

inDX

The horizontal distance to move the view. Negative values move the view to the left, positive values to the right.

inDY

The vertical distance to move the view. Negative values move the view up, positive values down.

Return Value
An operating system result code.

Discussion
This function affects the view's frame but does not affect the view's bounds.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewNewTrackingArea

Creates a new tracking area for a view.

```
OSStatus HIViewNewTrackingArea (
   HIViewRef inView,
   HIShapeRef inShape,
   HIViewTrackingAreaID inID,
   HIViewTrackingAreaRef *outRef
);
```

Parameters

inView

The view for which a new tracking area is to be created.

inShape

The shape to use. Pass `NULL` to use the entire structure region of the view. On return, you may safely release the shape.

inID

An identifier for the new tracking area. You can specify any value you want, or zero if you don't want to associate an identifier with the new tracking area.

outRef

On return, a reference to the new tracking area. In Mac OS X v10.5 and later, you may pass `NULL` if you don't need this information. The new tracking area is automatically destroyed when the view is released; you do not need to dispose of the tracking area yourself unless you remove it from the view before the view is released.

Return Value
An operating system result code.

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

Related Sample Code
CarbonCocoa_PictureCursor

Declared In
`HIView.h`

HIViewPlaceInSuperviewAt

Places a view at an absolute location within its parent.

```
OSStatus HIViewPlaceInSuperviewAt (
   HIViewRef inView,
   CGFloat inX,
   CGFloat inY
);
```

Parameters

inView

The view you want to position.

inX

The absolute horizontal coordinate at which to position the view.

inY

The absolute vertical coordinate at which to position the view.

Return Value
An operating system result code.

Discussion
This function affects the view's frame but does not affect the view's bounds.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewRegionChanged

Informs the system that a region of the view has changed.

```
OSStatus HIViewRegionChanged (
   HIViewRef inView,
   HIViewPartCode inRegionCode
);
```

Parameters

inView

The view whose region changed.

inView

The region that changed. At present, the region can only be the structure, opaque, and clickable regions. For possible constants, see "HIView Meta-Parts Constants" (page 124).

Return Value

An operating system result code.

Discussion

The view system may respond to the information provided by this function in some way. For example, if a view's clickable region changes, call this function to tell the Toolbox to resynchronize the region it uses for asynchronous window dragging (if enabled). Likewise, if a view's opaque region changes, call this function to adjust the window's opaque shape.

You don't need to call this function when a view is moved or resized because the HIToolbox automatically handles those kinds of changes.

Availability

Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In

HIView.h

HIViewRemoveFromSuperview

Removes a view from its parent.

```
OSStatus HIViewRemoveFromSuperview (
   HIViewRef inView
);
```

Parameters

inView

The view to remove.

Return Value

An operating system result code.

Discussion

The subview that is removed from the parent is not released and still exists.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Related Sample Code
QTCarbonShell

Declared In
`HIView.h`

HIViewRender

Renders the invalid portions of a view.

```
OSStatus HIViewRender (
   HIViewRef inView
);
```

Parameters

inView

The view that is to be rendered.

Return Value
An operating system result code.

Discussion
Normally, areas are redrawn at event loop time, but there might be times when an immediate redraw is needed. You should call this function sparingly because it does a fully composited redraw for the area of the view. That is, all other views that intersect the area of the specified view are also redrawn. Calling this function for several views at a particular level of a hierarchy would be costly, so you should only pass the root view of a window to this function.

The behavior of this function when passed a non-root view changed in Mac OS X v10.4. In Mac OS X v10.3, when called on a non-root view, this function validated all of the views in the window that intersect the specified view, including portions that did not intersect the specified view. Consequently, all of the views were not actually redrawn. In Mac OS X v10.4, when called on a non-root view, this function only validates those portions of each view that intersect the specified view.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewReshapeStructure

Informs the system that the structure region of the given view has changed shape.

```
OSStatus HIViewReshapeStructure (
   HIViewRef inView
);
```

Parameters

inView

The view to reshape and invalidate.

Return Value
An operating system result code.

Discussion
This function is used by custom views. If a view decides that its structure will change shape, it should call this function. This tells the toolbox to recalculate and invalidate as appropriate. You might, for example, call this function when gaining or losing a focus ring.

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

Not available to 64-bit applications.

Declared In
HIView.h

HIViewResumeLayout

Resumes layout handling for a view and its children.

```
OSStatus HIViewResumeLayout (
   HIViewRef inView
);
```

Parameters

inView

The view for which layout handling is to be resumed.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.

Not available to 64-bit applications.

Declared In
HIView.h

HIViewScrollRect

Scrolls a view's contents, or a portion thereof.

```
OSStatus HIViewScrollRect (
   HIViewRef inView,
   const HIRect *inRect,
   CGFloat inDX,
   CGFloat inDY
);
```

Parameters

inView

The view to scroll.

inRect

The rect to scroll. Pass `NULL` to mean the entire view. The rect passed cannot be bigger than the view's bounds. It must be in the local coordinate system of the view.

inDX

The horizontal distance to scroll. Positive values shift to the right, negative values shift to the left.

inDY

The vertical distance to scroll. Positive values shift downward, negative values shift upward.

Return Value
An operating system result code.

Discussion
This function blits the contents of the view as appropriate to scroll and then invalidates those portions that need to be redrawn. Be warned that this is a raw bit scroll. Anything that overlaps the target view will be scrolled as well.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSetActivated

Sets a view to be active or inactive.

```
OSStatus HIViewSetActivated (
   HIViewRef inView,
   Boolean inSetActivated
);
```

Parameters

inView

The view that is to be set.

inSetActivated

A Boolean whose value is `true` to set the view to be active or `false` to set the view to be inactive.

Return Value
An operating system result code.

Discussion
This function affects the effective activation of a view, which affects the effective activation of the view's children. If the view is being set to inactive, all children become inactive as well, but their latent activation does not change. If the children are latently inactive and the view is made active, the children remain latently inactive.

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

Declared In
`HIView.h`

HIViewSetBoundsOrigin

Sets the origin of the view.

```
OSStatus HIViewSetBoundsOrigin (
   HIViewRef inView,
   CGFloat inX,
   CGFloat inY
);
```

Parameters

inView
The view whose origin you wish to adjust.

inX
The X coordinate.

inY
The Y coordinate.

Return Value
An operating system result code.

Discussion
This effectively also moves all subcontrols of a view as well. This call will not invalidate the view, in case you might want to move the contents with `HIViewScrollRect` instead of redrawing the complete content.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
`HIView.h`

HIViewSetCommandID

Sets the command ID of a view.

```
OSStatus HIViewSetCommandID (
   HIViewRef inView,
   UInt32 inCommandID
);
```

Parameters

inView

The view whose command ID is to be set.

inID

The command ID to set.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewSetDrawingEnabled

Turns control drawing on or off.

```
OSStatus HIViewSetDrawingEnabled (
   HIViewRef inView,
   Boolean inEnabled
);
```

Parameters

inView

The view to enable or disable drawing for.

inEnabled

A Boolean value indicating whether drawing should be on (`true`) or off (`false`).

Return Value
An operating system result code.

Discussion
You can use this function to ensure that no drawing events are sent to the specified control. Functions such as `Draw1Control` and `HIViewSetNeedsDisplay` cannot draw when control drawing is off.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSetEnabled

Enables or disables a view.

```
OSStatus HIViewSetEnabled (
   HIViewRef inView,
   Boolean inSetEnabled
);
```

Parameters

inView

The view that is to be set.

inSetEnabled

A Boolean whose value is `true` to enable the view or `false` to disable the view.

Return Value
An operating system result code.

Discussion
Any subviews of the view specified by `inView` become enabled or disabled in accordance with the value of `inSetEnabled`.

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

Declared In
`HIView.h`

HIViewSetFirstSubViewFocus

Sets the subview that is first to receive keyboard focus.

```
OSStatus HIViewSetFirstSubViewFocus (
   HIViewRef inParent,
   HIViewRef inSubView
);
```

Parameters

inParent

The parent view.

inSubView

The first subview that is to receive keyboard focus. Pass `NULL` to tell the view system to use the default rules.

Return Value
An operating system result code.

Discussion
This function sets the first subview to shift focus to whenever the keyboard focus is advanced and the container view is entered.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSetFrame

Sets the frame of a view.

```
OSStatus HIViewSetFrame (
   HIViewRef inView,
   const HIRect *inRect
);
```

Parameters

inView

The view whose frame is to be set.

inRect

The new frame to set.

Return Value

An operating system result code.

Discussion

This function effectively moves the view within its parent. It also marks the view (and anything that was exposed behind it) to be redrawn.

Availability

Available in Mac OS X v10.2 and later.

Not available to 64-bit applications.

Related Sample Code

HID Calibrator

Declared In

`HIView.h`

HIViewSetHilite

Sets highlighting on a view.

```
OSStatus HIViewSetHilite (
   HIViewRef inView,
   HIViewPartCode inHilitePart
);
```

Parameters

inView

The view for which highlighting is to be set.

inHilitePart

The part of the view whose highlighting is to be set. For possible values, see "HIViewPartCode Constants" (page 126).

Return Value

An operating system result code.

Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In
HIView.h

HIViewSetID

Sets the HIViewID of a view.

```
OSStatus HIViewSetID (
   HIViewRef inView,
   HIViewID inID
);
```

Parameters

inView
The view whose HIViewID is to be set.

inID
The HIViewID to set.

Return Value
An operating system result code.

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

Declared In
HIView.h

HIViewSetLayoutInfo

Sets the layout information of an HIView.

```
OSStatus HIViewSetLayoutInfo (
   HIViewRef inView,
   const HILayoutInfo *inLayoutInfo
);
```

Parameters

inView
The view whose layout information is to be set.

inLayoutInfo
A pointer to an HILayoutInfo (page 106) structure containing the layout values that are to be set.

Return Value
An operating system result code.

Discussion
Layouts are used to automatically change the size and positioning of a view when another view changes size or position. Layout changes only take effect in two cases:

- When HIViewApplyLayout (page 56) is called after the view's layout has been set by calling HIViewSetLayoutInfo. The HIViewApplyLayout function is most useful when first applying a scaling or positioning layout to a view in order to set up the view's initial position and size relative to the view referenced in the layout.
- When the related view changes its size or position.

A layout allows a view's size or position to be modified in three ways: side binding, axial scaling, and axial positioning. In the following examples, View A initially has a left side of 100 and a right side of 150, and is placed inside a parent view that has a width of 275.

- Bindings — Setting up a binding relationship between two views specifies that one edge of a view is to change by an amount equal to the change in an edge of another view. If View A has a right-side binding to its parent's maximum (or right) side, when the parent's right side changes, View A's right side changes by the same amount. If the parent view resizes to be 325 units wide (50 units wider than before), View A is resized so that its left side changes to 150 units and its right side changes to 200, which is 50 units more to the right than before. You can think of bindings as a way to maintain distance. An edge bound to another view's edge always maintains its offset from that related edge. In other view systems, this concept is often referred to as "springs and rods." Note that a binding does *not* cause one side of the view to exactly align with the side of another view; a binding merely causes one side of the view to change by the same amount as another view changes. To align one edge of a view to another view's edge, use positioning. Bindings depend on changes in size or position of the related view. As a result, calling HIViewApplyLayout does not activate side bindings, as no changes have occurred. Bindings are implemented using the HIBinding (page 106) structure and one HISideBinding (page 107) structure per view edge.
- Scaling — Setting up a scaling relationship between two views specifies that the axial size (that is, the width or height) of a view is to be a specified ratio of the size of another view when that other view moves or resizes. If View A has an x-axis scaling for its parent view with a ratio of 0.8, when the parent view's width changes, View A's width changes to be the parent's width multiplied by 0.8. If the parent view resizes to be 325 units wide, View A resizes so that its left side stays at 100 and its right side changes to 360 (100 + 325 * 0.8). Note that when a scaling layout is first set up on a view with HIViewSetLayoutInfo, no scaling is applied to the view because scaling only occurs when the related view resizes. If scaling is required at initial setup, call HIViewSetLayoutInfo and then HIViewApplyLayout (page 56). Scaling is implemented using the HIScaling (page 107) structure and one HIAxisScale (page 107) structure per view axis.
- Positioning — Setting up a positioning relationship between two views specifies that the axial position (i.e., vertical or horizontal) of a view is to change so that the view aligns with the minimum, maximum, or center of another view when that other view resizes. If View A has an x-axis position with its parent view with center positioning specified, when the parent view changes size, View A moves so that it is centered horizontally relative to its parent. If the parent view resizes to be 300 units wide, View A repositions so that its left side is at 125 and its right side is at 175, centered in the parent view. Positioning is implemented using the HIPositioning (page 108) structure and one HIAxisPosition (page 108) per view axis.

The HIView layout engine applies transformations to a view sequentially. First, bindings are applied. Then scaling is applied, which could override some of the previously applied bindings. Then positioning is applied, which could also override some of the previously applied bindings. The bindings are applied recursively to a container's subviews, which requires care on your part to avoid infinite recursion, especially when applying inter-relational bindings. For example, if View A's x axis is scaled relative to View B and View B's x-axis is scaled to View A, your application could hang when the layouts are applied because View A would affect View B, which would affect View A, and so on.

For more information on using the layout engine, see *HIView Programming Guide*.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator

Declared In
`HIView.h`

HIViewSetMaximum

Sets a view's maximum value.

```
OSStatus HIViewSetMaximum (
   HIViewRef inView,
   SInt32 inMaximum
);
```

Parameters

inView
The view whose maximum value is to be set.

inView
The maximum value to set.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewSetMinimum

Sets a view's minimum value.

```
OSStatus HIViewSetMinimum (
   HIViewRef inView,
   SInt32 inMinimum
);
```

Parameters

inView
The view whose minimum value is to be set.

inMinimum
The value to set as the view's minimum value.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewSetNeedsDisplay

Marks a view as needing or not needing to be redrawn.

```
OSStatus HIViewSetNeedsDisplay (
   HIViewRef inView,
   Boolean inNeedsDisplay
);
```

Parameters

inView
The view to mark as dirty (needing to be redrawn) or clean (not needing to be redrawn).

inNeedsDisplay
A Boolean whose value is `true` to mark the view as dirty or `false` to mark it as clean.

Return Value
An operating system result code.

Discussion
If the view is not visible or is obscured completely by other views, no action is taken.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
HID Calibrator
HID Config Save
HID Explorer

Declared In
`HIView.h`

HIViewSetNeedsDisplayInRect

Uses an `HIRect` to mark a portion of a view as needing or not needing to be redrawn.

```
OSStatus HIViewSetNeedsDisplayInRect (
   HIViewRef inView,
   const HIRect *inRect,
   Boolean inNeedsDisplay
);
```

Parameters

inView

The view having a region that is to be marked as dirty (needs to be redrawn) or clean (valid and not needing to be redrawn).

inRect

The area, in view-relative coordinates, that is to be marked.

inNeedsDisplay

A Boolean whose value is `true` to mark the area described by `inRect` as dirty or `false` to mark it as clean.

Return Value
An operating system result code.

Discussion
If the view is not visible or is obscured completely by other views, no action is taken. The area specified by `inRect` is intersected with the view's visible region.

Availability
Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSetNeedsDisplayInRegion

Uses a region to mark a portion of a view as needing or not needing to be redrawn.

```
OSStatus HIViewSetNeedsDisplayInRegion (
   HIViewRef inView,
   RgnHandle inRgn,
   Boolean inNeedsDisplay
);
```

Parameters

inView

The view having a region that is to be marked as dirty (needs to be redrawn) or clean (valid and not needing to be redrawn).

inRgn

The region, in view-relative coordinates, to mark as dirty or clean.

inNeedsDisplay

A Boolean whose value is `true` to mark the region described by `inRgn` as dirty or `false` to mark it as clean.

Return Value
An operating system result code.

Discussion
If the view is not visible or is obscured completely by other views, no action is taken. The specified region is effectively intersected with the view's visible region.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSetNeedsDisplayInShape

Uses a shape to mark a portion of a view as needing or not needing to be redrawn.

```
OSStatus HIViewSetNeedsDisplayInShape (
   HIViewRef inView,
   HIShapeRef inArea,
   Boolean inNeedsDisplay
);
```

Parameters
`inView`
The view having a shape that is to be marked as dirty (needs to be redrawn) or clean (valid and not needing to be redrawn).
`inArea`
The area, in view-relative coordinates, that is to be marked.
`inNeedsDisplay`
A Boolean whose value is `true` to mark the area described by `inArea` as dirty or `false` to mark it as clean.

Return Value
An operating system result code.

Discussion
If the view is not visible or is obscured completely by other views, no action is taken. The area specified by `inArea` is intersected with the view's visible region.

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

Declared In
`HIView.h`

HIViewSetNextFocus

Sets the view that is to receive keyboard focus when keyboard focus advances from the specified view.

```
OSStatus HIViewSetNextFocus (
   HIViewRef inView,
   HIViewRef inNextFocus
);
```

Parameters

inView

The view.

inNextFocus

The view that is to receive keyboard focus when focus is advanced from the view specified by `inView`. The view must have the same parent as the view specified by `inView`. Pass `NULL` to tell the view system to use the default rules.

Return Value
An operating system result code.

Discussion
This function sets the view to which keyboard focus is to be shifted the next time keyboard focus is advanced from the view specified by `inView`.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSetText

Sets the text of a view to the specified string.

```
OSStatus HIViewSetText (
   HIViewRef inView,
   CFStringRef inText
);
```

Parameters

inView

The view whose text is to be set.

inText

The text that is to be set.

Return Value
An operating system result code.

Discussion
This function attempts to set the text that is displayed when drawing the view and is generally successful on views that handle the `kControlEditTextCFStringTag` `SetControlData` tag. If this function can't set that text, it sets the text in the view's title instead.

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

Related Sample Code
HID Explorer

Declared In
`HIView.h`

HIViewSetValue

Sets the value of a view.

```
OSStatus HIViewSetValue (
   HIViewRef inView,
   SInt32 inValue
);
```

Parameters

inView
The view whose value is to be set.

inValue
The value to set.

Return Value
An operating system result code.

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

Declared In
`HIView.h`

HIViewSetViewSize

Sets the view size of a view.

```
OSStatus HIViewSetViewSize (
   HIViewRef inView,
   SInt32 inViewSize
);
```

Parameters

inView
The view whose view size is to be set.

inViewSize
The view size that is to be set.

Return Value
An operating system result code.

Discussion
The view size is the size of the content to which a view's display is proportioned. The view size is commonly used to set the proportional size of a scroll bar's thumb indicator.

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

Declared In
`HIView.h`

HIViewSetVisible

Hides or shows a view.

```
OSStatus HIViewSetVisible (
   HIViewRef inView,
   Boolean inVisible
);
```

Parameters

inView
The view to hide or show.

inVisible
A Boolean value that indicates whether you want to hide the view (`false`) or show the view (`true`).

Return Value
An operating system result code.

Discussion
Marks the area the view will occupy or previously occupied as needing to be redrawn later.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Related Sample Code
CarbonCocoa_PictureCursor
HID Calibrator

Declared In
`HIView.h`

HIViewSetZOrder

Changes the front-to-back ordering of sibling views.

```
OSStatus HIViewSetZOrder (
   HIViewRef inView,
   HIViewZOrderOp inOp,
   HIViewRef inOther
);
```

Parameters

inView
The view whose Z-order you want to change.

inOp

Causes `inView` to be ordered above or below `inOther`. For possible values, see "HIView Z-Ordering Constants" (page 124).

inOther

Another optional view to use as a reference. Pass `NULL` to indicate an absolute position.

Return Value
An operating system result code.

Discussion
For example, passing `kHIViewZOrderAbove` as the value on `inOp` and `NULL` as the value of `inOther` moves a view to the front of all of its siblings. Passing `kHIViewZOrderBelow` as the value of `inOp` and `NULL` as the value of `inOther` moves a view to the back of all its siblings.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSimulateClick

Simulates a mouse click on a given view.

```
OSStatus HIViewSimulateClick (
   HIViewRef inView,
   HIViewPartCode inPartToClick,
   UInt32 inModifiers,
   HIViewPartCode *outPartClicked
);
```

Parameters

inView

The view to test the part hit.

inPartToClick

The part the view should consider to be clicked.

inModifiers

The modifiers the view can consider for its click action.

outPartClicked

The part that was hit; can be `kHIViewNoPart` if no action occurred. For possible values, see "HIViewPartCode Constants" (page 126). Pass `NULL` if you don't need the part code returned.

Return Value
An operating system result code.

Discussion
This function is used to simulate a mouse click on a given view. It sends a `kEventControlSimulateHit` event to the specified view and also sends `kEventControlHit` and (if the Hit event is not handled) `kEventCommandProcess` events.

Note that not all windows respond to the events that this API sends. A fully Carbon-event-based window most likely responds exactly as if the user had really clicked in the view. A window that uses Classic event record-based APIs (`WaitNextEvent` or `ModalDialog`) typically does not respond at all. To simulate a click in such a window, you may need to post a mouse-down/mouse-up pair or use a Dialog Manager event filter proc to simulate a hit in a dialog item.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSubtreeContainsFocus

Determines whether a view or any subviews have keyboard focus.

```
Boolean HIViewSubtreeContainsFocus (
   HIViewRef inSubtreeStart
);
```

Parameters

inSubtreeStart
The view to query.

Return Value
A Boolean whose value is `true` if the view or any of its children have keyboard focus; otherwise, `false`.

Availability
Available in Mac OS X v10.2 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

HIViewSuspendLayout

Suspends layout handling for a view and its children.

```
OSStatus HIViewSuspendLayout (
   HIViewRef inView
);
```

Parameters

inView
The view for which layout handling is to be suspended.

Return Value
An operating system result code.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`HIView.h`

Data Types

HILayoutInfo

Structure that stores the layout of an HIView.

```
struct HILayoutInfo {
    UInt32 version;
    HIBinding binding;
    HIScaling scale;
    HIPositioning position;
};
```

Fields

`version`
The version of this structure. The current version is `kHILayoutInfoVersionZero`.

`binding`
An `HIBinding` structure describing the bindings to apply to the sides of an HIView.

`scale`
An `HIScaling` structure describing the axial scaling to apply to an HIView.

`position`
An `HIPositioning` structure describing the positioning to apply to an HIView.

Discussion
This structure is provided as a parameter to `HIViewGetLayoutInfo` (page 71) and `HIViewSetLayoutInfo` (page 95).

HIBinding

Represents a set of top, left, bottom, and right bindings for an view.

```
struct HIBinding {
    HISideBinding top;
    HISideBinding left;
    HISideBinding bottom;
    HISideBinding right;
};
```

Fields

`top`
The top side bindings.

`left`
The left side bindings.

`bottom`
The bottom side bindings.

`right`
The right side bindings.

Discussion
These constants are used in conjunction with the HIView layout engine.

HISideBinding

Structure for storing the binding for the side of a view.

```
struct HISideBinding {
    HIViewRef toView;
    HIBindingKind kind;
    float offset;
};
```

Fields
`toView`
An `HIViewRef` to the view to which this side is bound. This field can be `NULL`, which indicates that the side is bound to its parent view.

`kind`
The bind kind. For possible values, see "HILayout Binding Kind Constants" (page 118).

`offset`
Reserved; must be set to zero.

Discussion
The layout engine can automatically reposition and resize views for which relationships have been set up. (Call `HIViewSetLayoutInfo` (page 95) to establish these relationships.) A side binding is entirely related to the change of the parent's position or size but only as the size affects the maximum edge position. A side binding doesn't mean "move to where my relative's side is" but rather "move as my relative's side has moved."

HIScaling

A set of scaling descriptions for the axes of a view.

```
struct HIScaling {
    HIAxisScale x;
    HIAxisScale y;
};
```

Fields
`x`
The horizontal scaling for a view.

`x`
The vertical scaling for a view.

HIAxisScale

Represents a scale description for an axis of a view.

```
struct HIAxisScale {
    HIViewRef toView;
    HIScaleKind kind;
    float ratio;
};
```

Fields

`toView`

An `HIViewRef` to the view to which this axis is scaled. This field can be `NULL`, which indicates that the axis is scaled relative to its parent's view.

`kind`

The type of scaling. Currently, this field must be `kHILayoutScaleAbsolute`.

`ratio`

A value that indicates how much to scale the view. Zero indicates no scaling. A value of one indicates that the view is to always have the same axial size.

HIPositioning

A positioning description for an HIView.

```
struct HIPositioning {
    HIAxisPosition x;
    HIAxisPosition y;
};
```

Fields

`x`

The X axis.

`y`

The Y axis.

HIAxisPosition

An axial position description for an HIView.

```
struct HIAxisPosition {
    HIViewRef toView;
    HIPositionKind kind;
    float offset;
};
```

Fields

`toView`

An `HIViewRef` to the view relative to which a view positioned. This field can be `NULL`, which indicates that the axis is positioned relative to its parent's view.

`kind`

The type of positioning. For possible values, see "HIPositionKind Constants" (page 119).

`offset`

After the position kind has been applied, the origin component that corresponds to the positioning axis is offset by this value. For example, left aligned + 10.

HIViewContentInfo

Describes the content of a view.

```
struct HIViewContentInfo {
    HIViewContentType contentType;
    union {
        IconSuiteRef iconSuite;
        IconRef iconRef;
        CGImageRef imageRef;
    } u;
};
typedef struct HIViewContentInfo HIViewContentInfo;
typedef HIViewContentInfo * HIViewContentInfoPtr;
```

Fields

`contentType`

The type of content in the union. For possible values, see "HIViewContentType Constants" (page 125).

Availability

Available in Mac OS X v10.4 and later.

Declared In

`HIView.h`

HIViewID

Defines the HIView ID.

```
typedef ControlID HIViewID;
```

Availability

Available in Mac OS X v10.2 and later.

Declared In

`HIView.h`

HIViewFrameMetrics

Describes the offsets from the structure to the content area of a view.

```
struct HIViewFrameMetrics {
    float top;
    float left;
    float bottom;
    float right;
};
```

Fields

`top`

Height of the top of the structure area.

`left`

Width of the left of the structure area.

`bottom`
Height of the bottom of the structure area.

`right`
Width of the right of the structure area.

Discussion
The top metric is the difference between the vertical coordinate of the top edge of the view's structure region and the vertical coordinate of the top edge of the view's content region. This structure is returned by a view in response to a `kEventControlGetFrameMetrics` event.

HIViewKind

Represents the signature and kind of the view.

```
struct HIViewKind {
    OSType signature;
    OSType kind;
}
```

Fields
`signature`
The signature of the view. Apple reserves all signatures made up of only lowercase characters.

`kind`
The kind of the view. Apple reserves all kinds made up of only lowercase characters.

HIViewRef

Define an HIView reference.

```
typedef ControlRef HIViewRef;
```

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

Declared In
`HIObject.h`

HIViewTrackingAreaRef

Define an HIView tracking area reference.

```
typedef struct OpaqueHIViewTrackingAreaRef HIViewTrackingAreaRef;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`HIView.h`

HIViewTrackingAreaID

Define an HIView tracking area ID.

```
typedef UInt64 HIViewTrackingAreaID;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`HIView.h`

Constants

Class ID Constants

Specify class identifiers for view-related subclasses of HIObject.

```
#define kHIViewClassID              CFSTR("com.apple.hiview")
#define kHIGrowBoxViewClassID       CFSTR("com.apple.higrowboxview")
#define kHIImageViewClassID         CFSTR("com.apple.HIImageView")
#define kHIMenuViewClassID          CFSTR("com.apple.HIMenuView")
#define kHIStandardMenuViewClassID  CFSTR("com.apple.HIStandardMenuView")
#define kHISegmentedViewClassID     CFSTR("com.apple.HISegmentedView")
#define kHIScrollViewClassID        CFSTR("com.apple.HIScrollView")
#define kHIComboBoxClassID          CFSTR("com.apple.HIComboBox")
#define kHISearchFieldClassID       CFSTR("com.apple.HISearchField")
#define kHICocoaViewClassID         CFSTR("com.apple.HICocoaView")
```

Constants

`kHIViewClassID`
Class identifier for the HIView class.
Available in Mac OS X v10.2 and later.
Declared in `HIView.h`.

`kHIGrowBoxViewClassID`
Class identifier for the HIGrowBoxView class.
Available in Mac OS X v10.2 and later.
Declared in `HIWindowViews.h`.

`kHIImageViewClassID`
Class identifier for the HIImageView class.
Available in Mac OS X v10.3 and later.
Declared in `HIImageViews.h`.

`kHIMenuViewClassID`
Class identifier for the HIMenuView class.
Available in Mac OS X v10.3 and later.
Declared in `HIMenuView.h`.

`kHIStandardMenuViewClassID`
Class identifier for the HIStandardMenuView class.

Available in Mac OS X v10.3 and later.

Declared in `HIMenuView.h`.

`kHISegmentedViewClassID`
Class identifier for the HISegmentedView class.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

`kHIScrollViewClassID`
Class identifier for the HIScrollView class.

Available in Mac OS X v10.3 and later.

Declared in `HIScrollView.h`.

`kHIComboBoxClassID`
Class identifier for the HIComboBox class.

Available in Mac OS X v10.3 and later.

Declared in `HIComboBox.h`.

`kHISearchFieldClassID`
Class identifier for the HISearchField class.

Available in Mac OS X v10.3 and later.

Declared in `HISearchField.h`.

`kHICocoaViewClassID`
Class identifier for the HICocoaView class.

Available in Mac OS X v10.5 and later.

Declared in `HICocoaView.h`.

Clock Event Constant

Specify the constant for changes in date or time in the clock control (`kEventClassClockView`).

```
enum {
    kEventClockDateOrTimeChanged = 1
};
```

Constants

`kEventClockDateOrTimeChanged`
An event sent by the clock control when the user changes the date or time. Clients can register for this notification in order to update state based on the date or time in the clock. This event is sent to the view only; it is not propagated. The event is sent to all handlers installed on the control.

Available in Mac OS X v10.4 and later.

Declared in `HIClockView.h`.

Discussion
Table 1 shows the event parameters associated with this event.

Table 1 Parameter names and types for date or time change events

Event kind	Parameter name	Parameter type

kEventClockDateOrTimeChanged	kEventParamDirectObject	typeControlRef

Combo Box Attributes

Specify constants for combo box attributes.

```
enum {
    kHIComboBoxNoAttributes = 0L,
    kHIComboBoxAutoCompletionAttribute = (1L << 0),
    kHIComboBoxAutoDisclosureAttribute = (1L << 1),
    kHIComboBoxAutoSortAttribute = (1L << 2),
    kHIComboBoxAutoSizeListAttribute = (1L << 3),
    kHIComboBoxStandardAttributes = (kHIComboBoxAutoCompletionAttribute |
kHIComboBoxAutoDisclosureAttribute | kHIComboBoxAutoSizeListAttribute)
};
```

Constants

`kHIComboBoxNoAttributes`

Indicates the lack of any attributes.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxAutoCompletionAttribute`

The control will attempt to auto complete the text the user is typing with an item in the combo box list that is the closest appropriate match.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxAutoDisclosureAttribute`

The control will disclose the combo box list after the user enters text.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxAutoSortAttribute`

The items in the combo box list will be automatically sorted in alphabetical order.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxAutoSizeListAttribute`

The combo box list will be automatically sized to fit the Human Interface Guidelines.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxStandardAttributes`

The minimum set of commonly used combo box attributes.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

Combo Box Data Tags

Specify constants for combo box data tags.

```
enum {
    kHIComboBoxListTag = 'cbls',
    kHIComboBoxListPixelWidthTag = 'cblw',
    kHIComboBoxListPixelHeightTag = 'cblh',
    kHIComboBoxNumVisibleItemsTag = 'cbni'
};
```

Constants

`kHIComboBoxListTag`

Extract the contents of the combo box list as a `CFArray`. The `CFArray` is retained; if you get the array, you own it and must release it. If you set it by calling `SetControlData`, the toolbox makes a copy of it, and you are free to release your reference. The reference count is bumped on get/retains on set.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxListPixelWidthTag`

A `UInt32` containing the width of the combo box list. The width can be customized. This tag disables the auto size attribute.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxListPixelHeightTag`

A `UInt32` containing the height of the combo box list. The height can be customized. This tag disables the auto size attribute.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

`kHIComboBoxNumVisibleItemsTag`

A `UInt32` containing the number of visible items in the combo box list. The number can be customized. This tag disables the auto size attribute.

Available in Mac OS X v10.2 and later.

Declared in `HIComboBox.h`.

Discussion

The combo box view also supports these tags previously defined for the `EditUnicodeText` control. These tags are available through `GetControlData` and `SetControlData` with a `ControlPartCode` of `kHIComboBoxEditTextPart`:

- `kControlFontStyleTag`
- `kControlEditTextFixedTextTag`
- `kControlEditTextTextTag`
- `kControlEditTextKeyFilterTag`
- `kControlEditTextValidationProcTag`
- `kControlEditUnicodeTextPostUpdateProcTag`
- `kControlEditTextSelectionTag`
- `kControlEditTextKeyScriptBehaviorTag`

- kControlEditTextCharCount
- kControlEditTextCFStringTag

Combo Box List Item Event Constants

Specify a constant for combo box list item events (kEventClassHIComboBox).

```
enum {
    kEventComboBoxListItemSelected = 1,
    kEventParamComboBoxListSelectedItemIndex = 'cbli'
};
```

Constants

kEventComboBoxListItemSelected

This event is sent as a notification when an item in the combo box disclosure list has been selected. This event is sent to all handlers installed on the control. This does not imply that the selection has been accepted; for that you will need to register for the kEventClassTextField/kEventTextAccepted event.

Available in Mac OS X v10.4 and later.

Declared in HIComboBox.h.

kEventParamComboBoxListSelectedItemIndex

Indicates the index of a selected combo box list item.

Available in Mac OS X v10.4 and later.

Declared in HIComboBox.h.

Discussion

Table 2 shows the event parameters associated with these events.

Table 2 Parameter names and types for combo box events

Event kind	Parameter name	Parameter type
kEventComboBoxList-ItemSelected	kEventParamDirectObject	typeControlRef
	kEventParamComboBoxListItemSelected-ItemIndex	typeCFIndex

Combo Box Part Constants

Specify constants for combo box part codes.

```
enum {
    kHIComboBoxEditTextPart = 5,
    kHIComboBoxDisclosurePart = 28
};
```

Constants

`kHIComboBoxEditTextPart`
Edit text part.
Available in Mac OS X v10.2 and later.
Declared in `HIComboBox.h`.

`kHIComboBoxDisclosurePart`
Disclosure part.
Available in Mac OS X v10.2 and later.
Declared in `HIComboBox.h`.

Discussion

Combo box part code constants are used when calling `HIViewSetHilite` (page 94), `HIViewCopyShape` (page 60) and are returned by `HIViewRegionChanged` (page 87), `HIViewGetFocusPart` (page 68). They are also used by the hit testing functions, `HIViewSimulateClick` (page 104) and `HIViewGetPartHit` (page 74).

Control Kind Constants

Specify constants for control kinds.

```
enum {
    kControlKindHIScrollView = 'scrl',
    kControlKindHIImageView = 'imag',
    kControlKindHIComboBox = 'cbbx',
    kControlKindHISearchField = 'srfd',
    kControlKindHIMenuView = 'menu',
    kControlKindHIStandardMenuView = 'smnu',
    kHISegmentedViewKind = 'sgmt',
    kControlKindHIGrowBoxView = 'grow',
    kControlKindHICocoaView = 'hins'
};
```

Constants

`kControlKindHIScrollView`
Control kind for a scroll view.
Available in Mac OS X v10.4 and later.
Declared in `HIScrollView.h`.

`kControlKindHIImageView`
Control kind for an image view.
Available in Mac OS X v10.4 and later.
Declared in `HIImageViews.h`.

`kControlKindHIComboBox`
Control kind for a combo box.
Available in Mac OS X v10.2 and later.
Declared in `HIComboBox.h`.

`kControlKindHISearchField`
Control kind for a search field.

Available in Mac OS X v10.3 and later.

Declared in `HISearchField.h`.

`kControlKindHIMenuView`
Control kind for a menu view.

Available in Mac OS X v10.4 and later.

Declared in `HIMenuView.h`.

`kControlKindHIStandardMenuView`
Control kind for a standard menu view.

Available in Mac OS X v10.4 and later.

Declared in `HIMenuView.h`.

`kHISegmentedViewKind`
Control kind for a segmented view.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

`kControlKindHIGrowBoxView`
Control kind for a grow box view.

Available in Mac OS X v10.4 and later.

Declared in `HIWindowViews.h`.

`kControlKindHICocoaView`
Control kind for a view that wraps a Cocoa view.

Available in Mac OS X v10.5 and later.

Declared in `HICocoaView.h`.

Discussion
These constants are returned by `HIViewGetKind` (page 70) and `GetControlKind`.

Event Class Constants

Specify event class constants.

```
enum {
    kEventClassClockView = 'cloc',
    kEventClassScrollable = 'scrl',
    kEventClassHIComboBox = 'hicb',
    kEventClassSearchField = 'srfd',
    kEventClassTextField = 'txfd'
};
```

Constants
`kEventClassClockView`
Event class for events related to a clock view.

Available in Mac OS X v10.4 and later.

Declared in `HIClockView.h`.

`kEventClassScrollable`
Event class for events related to a scrollable view.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kEventClassHIComboBox`
Event class for events related to a combo box view.

Available in Mac OS X v10.4 and later.

Declared in `HIComboBox.h`.

`kEventClassSearchField`
Event calls for events related to a search field view.

Available in Mac OS X v10.3 and later.

Declared in `HISearchField.h`.

`kEventClassTextField`
Event class for events related to text field views.

Available in Mac OS X v10.3 and later.

Declared in `HITextViews.h`.

HILayout Binding Kind Constants

Specify binding constants used by the HIView layout engine.

```
enum {
    kHILayoutBindNone = 0,
    kHILayoutBindMin = 1,
    kHILayoutBindMax= 2,
    kHILayoutBindLeft = kHILayoutBindMin,
    kHILayoutBindRight = kHILayoutBindMax,
    kHILayoutBindTop = kHILayoutBindMin,
    kHILayoutBindBottom = kHILayoutBindMax
};
typedef UInt16 HIBindingKind;
```

Constants

`kHILayoutBindNone`
No binding is to occur.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

`kHILayoutBindMin`
Bind to the minimum of the axis.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

`kHILayoutBindMax`
Bind to the maximum of the axis.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

`kHILayoutBindLeft`
Synonym for `kHILayoutBindMin`.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

`kHILayoutBindRight`
Synonym for `kHILayoutBindMax`.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

`kHILayoutBindTop`
Synonym for `kHILayoutBindMin`.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

`kHILayoutBindBottom`
Synonym for `kHILayoutBindMax`.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

Discussion
Mac OS X v10.3 provides a layout engine for HIViews that allows applications to specify the layout relationships between multiple views. When necessary, the layout engine automatically repositions and resizes views that have layout information. For more information on how to use the layout engine, see `HIViewSetLayoutInfo` (page 95).

Horizontal and vertical bindings are very similar in application, but along different axes, so the binding kinds have been abstracted to minimum and maximum. Synonyms are provided for convenience, and you are encouraged to use them.

HILayoutInfoVersion Constant

Specify version `0` of the `HILayoutInfo` structure

```
enum {
    kHILayoutInfoVersionZero = 0
};
```

Constants
`kHILayoutInfoVersionZero`
The version of the `HILayoutInfo` structure is `0`.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

HIPositionKind Constants

Specify position constants used by the HIView layout engine.

```
enum {
    kHILayoutPositionNone = 0,
    kHILayoutPositionCenter = 1,
    kHILayoutPositionMin = 2,
    kHILayoutPositionMax = 3,
    kHILayoutPositionLeft = kHILayoutPositionMin,
    kHILayoutPositionRight = kHILayoutPositionMax,
    kHILayoutPositionTop = kHILayoutPositionMin,
    kHILayoutPositionBottom = kHILayoutPositionMax
};
typedef UInt16 HIPositionKind;
```

Constants

`kHILayoutPositionNone`
No positioning is to occur.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionCenter`
Bind to the center.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionMin`
Bind to the minimum of the axis.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionMax`
Bind to the maximum of the axis.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionLeft`
Synonym for `kHILayoutPositionMin`.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionRight`
Synonym for `kHILayoutPositionMax`.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionTop`
Synonym for `kHILayoutPositiondMin`.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

`kHILayoutPositionBottom`
Synonym for `kHILayoutPositionMax`.
Available in Mac OS X v10.3 and later.
Declared in `HIView.h`.

HIScaleKind Constant

Specify a constant that indicates the scale is determined from the axis size.

```
enum {
    kHILayoutScaleAbsolute = 0
};
typedef UInt16 HIScaleKind;
```

Constants

`kHILayoutScaleAbsolute`

Indicates that the scale is determined from the axis size.

Available in Mac OS X v10.3 and later.

Declared in `HIView.h`.

Discussion

This constant is used in conjunction with the HIView layout engine.

HIView Attributes

Specify constants that change the behavior of controls.

```
enum {
    kHIViewAttributeSendCommandToUserFocus = 1 << 0,
    kHIViewAttributeIsFieldEditor = 1 << 1,
    kHIViewSendCommandToUserFocus = kHIViewAttributeSendCommandToUserFocus
};
```

Constants

`kHIViewAttributeSendCommandToUserFocus`

When set, the control sends the command it generates to the user focus; the command propagates as it would naturally from there. (The default is to send the command to itself and then to its parent and so forth.) You may want to use this setting for views that are not typically in a focused window. For example, a push button in a toolbar window might use this setting to cause its command to be sent to the focused window rather than to the toolbar window.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewAttributeIsFieldEditor`

Indicates that text editing controls should behave appropriately for editing fields in a dialog; specifically, the control should ignore the Return, Enter, Escape, and Tab keys, and allow them to be processed by other participants in the event flow.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewSendCommandToUserFocus`

Legacy constant. Use `kHIViewAttributeSendCommandToUserFocus` instead.

Available in Mac OS X v10.2 and later.

Declared in `HIView.h`.

Discussion

These constants are used by `HIViewGetAttributes` (page 65) to get a view's attributes and by `HIViewChangeAttributes` (page 56) to set or change a view's attributes.

HIView Feature Constants

Specify constants for view features.

```
enum {
    kHIViewFeatureSupportsGhosting = 1 << 0,
    kHIViewFeatureAllowsSubviews = 1 << 1,
    kHIViewFeatureGetsFocusOnClick = 1 << 8,
    kHIViewFeatureSupportsLiveFeedback = 1 << 10,
    kHIViewFeatureSupportsRadioBehavior = 1 << 11,
    kHIViewFeatureAutoToggles = 1 << 14,
    kHIViewFeatureIdlesWithTimer = 1 << 23,
    kHIViewFeatureInvertsUpDownValueMeaning = 1 << 24,
    kHIViewFeatureIsOpaque = 1 << 25,
    kHIViewFeatureDoesNotDraw = 1 << 27,
    kHIViewFeatureDoesNotUseSpecialParts = 1 << 28,
    kHIViewFeatureIgnoresClicks = 1 << 29
};
typedef UInt64 HIViewFeatures;
```

Constants

`kHIViewFeatureSupportsGhosting`

This view supports using the ghosting protocol when live tracking is not enabled. Use this constant instead of the legacy constant, `kHIViewSupportsGhosting`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureAllowsSubviews`

This view allows subviews to be embedded within it. Use this constant instead of the legacy constant, `kHIViewAllowsSubviews`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureGetsFocusOnClick`

If this view is clicked, the keyboard focus should be set to this view automatically; used primarily for edit text controls. Use this constant instead of the legacy constant, `kHIViewGetsFocusOnClick`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureSupportsLiveFeedback`

This view supports the live feedback protocol, which is necessary for implementing live scroll bar tracking. Clients of a view should never disable this bit. Use this constant instead of the legacy constant, `kHIViewSupportsLiveFeedback`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureSupportsRadioBehavior`

This view can be placed in a radio group. Radio buttons and bevel buttons report this behavior. Use this constant instead of the legacy constant, `kHIViewSupportsRadioBehavior`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureAutoToggles`

This view supports the auto-toggle protocol and should at the very least auto-toggle between off and on. The view can support a Carbon event for more advanced auto-toggling of its value. The tab view supports this, for example, so that when a tab is clicked its value changes automatically. Use this constant instead of the legacy constant, `kHIViewAutoToggles`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureIdlesWithTimer`

An informational bit. Turning this bit off would not necessarily disable any timer a view might be using, but a timer could obey this bit if desired. Use this constant instead of the legacy constant, `kHIViewIdlesWithTimer`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureInvertsUpDownValueMeaning`

This bit tells the Control Manager that when the Up button part of the control is clicked, the value of the control should increase. A Scroll bar, conversely, decreases in value when its Up button is clicked. Use this constant instead of the legacy constant, `kHIViewInvertsUpDownValueMeaning`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureIsOpaque`

When set, the view's structure region is used to determine the view's opaque region, and calling the view can usually be avoided. Use this constant instead of the legacy constant, `kHIViewIsOpaque`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureDoesNotDraw`

This bit is an optimization that tells the Control Manager that a view is transparent and does not do any drawing, so the Control Manager doesn't have to invalidate the view and instead should invalidate views behind this view. For example, on a metal window, the content view is actually fully transparent, so invalidating it is unnecessary. Use this constant instead of the legacy constant, `kHIViewDoesNotDraw`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureDoesNotUseSpecialParts`

Indicates to the Control Manager that this view doesn't use the special part codes for indicator, inactive, and disabled. Use this constant instead of the legacy constant, `kHIViewDoesNotUseSpecialParts`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewFeatureIgnoresClicks`

When set, this bit tells the Control Manager that it does not need to ask the control for it's clickable region. A view such as the visual separator would set this bit, and metal windows set this bit when doing asynchronous window dragging. This bit is typically set in conjunction with the `kHIViewFeatureDoesNotDrawBit`. Use this constant instead of the legacy constant, `kHIViewIgnoresClicks`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

Discussion
View feature flags are generally determined by the view itself and are not typically changed by clients of the view. Call `HIViewGetFeatures` (page 67) to obtain a view's features, and `HIViewChangeFeatures` (page 57) to set and clear a view's features.

HIView Meta-Parts Constants

Specify meta-parts constants used when calling `HIViewCopyShape`.

```
enum {
    kHIViewStructureMetaPart = -1,
    kHIViewContentMetaPart = -2,
    kHIViewOpaqueMetaPart = -3,
    kHIViewClickableMetaPart = -4
};
```

Constants

`kHIViewStructureMetaPart`
The structure region is the total area over which the view draws.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewContentMetaPart`
The content region is only defined by views that can embed other views. It is the area that embedded content can live.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewOpaqueMetaPart`
The portion of the view that is opaque. No views behind this portion of the view will be asked to draw because their drawing would be completely overwritten by this view's drawing.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewClickableMetaPart`
Used for asynchronous window dragging only. The default is the structure region.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

Discussion
Meta-parts are used when calling `HIViewCopyShape` (page 60). Meta-parts define a region of a view and they might be defined by a view. Along with these parts, you can also pass in normal part codes to get the regions of those parts. Not all views fully support this feature.

HIView Z-Ordering Constants

Specify constants that set a view's z-order.

```
enum {
    kHIViewZOrderAbove = 1,
    kHIViewZOrderBelow = 2,
};
typedef UInt32 HIViewZOrderOp;
```

Constants

`kHIViewZOrderAbove`
Order the view above another view.
Available in Mac OS X v10.2 and later.
Declared in `HIView.h`.

`kHIViewZOrderBelow`
Order the view below another view.
Available in Mac OS X v10.2 and later.
Declared in `HIView.h`.

Discussion
These constants are used when calling `HIViewSetZOrder` (page 103).

HIViewContentType Constants

Specify constants that describe a view's content.

```
enum {
    kHIViewContentTextOnly = 0,
    kHIViewContentNone = 0,
    kHIViewContentIconSuiteRef = 129,
    kHIViewContentIconRef = 132,
    kHIViewContentCGImageRef = 134
};
```

Constants

`kHIViewContentTextOnly`
The view has no content other than text.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewContentNone`
The view has no content.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewContentIconSuiteRef`
The view's content is an `IconSuiteRef`.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewContentIconRef`
The view's content is an `IconRef`.
Available in Mac OS X v10.4 and later.
Declared in `HIView.h`.

`kHIViewContentCGImageRef`
The view's content is an `CGImageRef`.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

Discussion
These constants are used in the `HIViewContentInfo` structure. For information on that structure, see HIViewContentInfo (page 109).

HIViewPartCode Constants

Specify view parts constants.

```
enum {
    kHIViewNoPart = 0,
    kHIViewIndicatorPart = 129,
    kHIViewDisabledPart = 254,
    kHIViewInactivePart = 255,
    kHIViewEntireView = kHIViewNoPart
};
typedef ControlPartCode HIViewPartCode;
```

Constants
`kHIViewNoPart`
The entire view; used when not referring to a specific part.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewIndicatorPart`
Indicator part.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewDisabledPart`
Disabled part.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewInactivePart`
Inactive part.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHIViewEntireView`
The entire view; used when not referring to a specific part.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

Discussion
These constants are used when calling `HIViewSetHilite` (page 94), `HIViewRegionChanged` (page 87), `HIViewCopyShape` (page 60), `HIViewSimulateClick` (page 104), `HIViewGetPartHit` (page 74), and `HIViewGetFocusPart` (page 68). These constants are also used with various `kEventClassControl` Carbon events.

Mouse Tracking Area Event Constants

Specify constants for mouse tracking area events (`kEventClassControl`).

```
enum {
    kEventControlTrackingAreaEntered = 23,
    kEventControlTrackingAreaExited = 24,
    kEventParamHIViewTrackingArea = 'ctra',
    typeHIViewTrackingAreaRef = 'ctra'
};
```

Constants

`kEventControlTrackingAreaEntered`

If you installed a mouse tracking area in your view, you will receive this event when the mouse enters that area. The tracking area reference is sent with the event.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kEventControlTrackingAreaExited`

If you installed a mouse tracking area in your view, you will receive this event when the mouse leaves that area. The tracking area reference is sent with the event.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kEventParamHIViewTrackingArea`

An `HIViewTrackingAreaRef` for the tracking area that was entered.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

Discussion

Table 3 shows the event parameters associated with these events.

Table 3 Parameter names and types for mouse tracking area events

Event kind	Parameter name	Parameter type
`kEventControl-TrackingAreaEntered`	`kEventParamHIViewTrackingArea`	`typeHIViewTrackingAreaRef`
	`kEventParamKeyModifiers`	`typeUInt32`
	`kEventParamMouseLocation`	`typeHIPoint`
`kEventControl-TrackingAreaExited`	`kEventParamHIViewTrackingArea`	`typeHIViewTrackingAreaRef`
	`kEventParamKeyModifiers`	`typeUInt32`
	`kEventParamMouseLocation`	`typeHIPoint`

Scroll View Constants

Specify constants for scroll view options.

```
enum {
    kHIScrollViewOptionsVertScroll = (1 << 0),
    kHIScrollViewOptionsHorizScroll = (1 << 1),
    kHIScrollViewOptionsAllowGrow = (1 << 2),
    kHIScrollViewValidOptions = (kHIScrollViewOptionsVertScroll |
kHIScrollViewOptionsHorizScroll | kHIScrollViewOptionsAllowGrow)
};
```

Constants

`kHIScrollViewOptionsVertScroll`
Indicates that a vertical scroll bar is desired.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kHIScrollViewOptionsHorizScroll`
Indicates that a horizontal scroll bar is desired.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kHIScrollViewOptionsAllowGrow`
Indicates that space for a grow box should be taken into account when laying out scroll bars. In Mac OS X v10.3 and earlier, if both horizontal and vertical scroll bars are requested, this attribute is assumed. In Mac OS X v10.4 and later, this attribute is *not* assumed; this allows the scroll view to support independent auto-hiding of the two scroll bars in Mac OS X v10.4 and later. If you want to preserve space for the grow box on all systems, specify this option bit.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kHIScrollViewValidOptions`
All valid scroll view options.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

Discussion

These constants are used in conjunction with `HIScrollViewCreate` (page 35).

Scroll View Action Constants

Specify constants for scroll view navigation actions.

```
enum {
    kHIScrollViewScrollToTop = (1 << 0),
    kHIScrollViewScrollToBottom = (1 << 1),
    kHIScrollViewScrollToLeft = (1 << 2),
    kHIScrollViewScrollToRight = (1 << 3),
    kHIScrollViewPageUp = (1 << 4),
    kHIScrollViewPageDown = (1 << 5),
    kHIScrollViewPageLeft = (1 << 6),
    kHIScrollViewPageRight = (1 << 7)
};
typedef UInt32 HIScrollViewAction;
```

Constants

`kHIScrollViewScrollToTop`
The scroll view should move to the top of the content.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewScrollToBottom`
The scroll view should move to the bottom of the content.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewScrollToLeft`
The scroll view should move to the left of the content.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewScrollToRight`
The scroll view should move to the right of the content.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewPageUp`
The scroll view should page up.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewPageDown`
The scroll view should page down.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewPageLeft`
The scroll view should page left.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

`kHIScrollViewPageRight`
The scroll view should page right.
Available in Mac OS X v10.3 and later.
Declared in `HIScrollView.h`.

Discussion
These constants are used in conjunction with `HIScrollViewNavigate` (page 37) and `HIScrollViewCanNavigate` (page 35).

Scrollable Event Constants

Specify constants for scrollable events (`kEventClassScrollable`).

```
enum {
    kEventScrollableGetInfo = 1,
    kEventScrollableInfoChanged = 2,
    kEventScrollableScrollTo = 10
};
```

Constants

`kEventScrollableGetInfo`
This event is sent by an HIScrollView to its scrollable view to determine the current size and origin of the scrollable view. A scrollable view must implement this event in order to scroll properly inside an HIScrollView.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kEventScrollableInfoChanged`
This event is not sent by an HIScrollView itself. Instead, it may be sent to an instance of HIScrollView to notify the scroll view that the size or origin of its scrollable view has changed. The HIScrollView responds to this event by sending a `kEventScrollableGetInfo` event to its scrollable view. It then updates the scroll bars appropriately to reflect the changes. It does *not* move the origin of the scrollable view at all. This event is just a notification to allow the scroll view to sync up with its scrollable view.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kEventScrollableScrollTo`
This event is sent by an HIScrollView to its scrollable view to request that the scrollable view update its current origin and redraw. Typically, a scrollable view scrolls its content by setting its bounds origin using `HIViewSetBoundsOrigin` (page 91) or by offsetting its drawing by the scroll origin. If the view embeds other views, it must use `HIViewSetBoundsOrigin` to allow the embedded views to scroll along with their containing view. A view that uses `HIViewSetBoundsOrigin` should call that API in response to this event. A view that offsets its drawing by the scroll origin should update its current origin in its own instance data in response to this event. A scrollable view should also use `HIViewScrollRect` (page 89) to scroll its content or `HIViewSetNeedsDisplay` (page 98) to cause itself to redraw using the new origin point. A scrollable view must implement this event in order to scroll properly inside an HIScrollView.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

Discussion
Table 4 shows the event parameters associated with these events.

Table 4 Parameter names and types for scrollable events

Event kind	Parameter name	Parameter type
`kEventScrollableGetInfo`	`kEventParamImageSize`	`typeHISize`

	kEventParamViewSize	typeHISize
	kEventParamLineSize	typeHISize
	kEventParamOrigin	typeHIPoint
kEventScrollableInfoChanged	kEventParamOrigin	typeHIPoint
kEventScrollableScrollTo	kEventParamOrigin	typeHIPoint

Scrollable Event Parameter Constants

Specify scrollable event parameter constants.

```
enum {
    kEventParamImageSize = 'imsz',
    kEventParamViewSize = 'vwsz',
    kEventParamLineSize = 'lnsz',
    kEventParamOrigin = 'orgn'
};
```

Constants

`kEventParamImageSize`

A value of type `typeHISize` representing the image size. The image size is the total size of the scrollable view, including any parts of the view that are not currently visible. For example, a scrollable view that displays a hundred page document would return an image size equal to one hundred times the height of the page.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kEventParamViewSize`

A value of type `typeHISize` representing the view size. The view size is the current size of the scrollable view. Typically, this is the same as the view's bounds and can be acquired by calling `HIViewGetBounds` (page 65).

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kEventParamLineSize`

A value of type `typeHISize` representing the line size. The line size is the distance that the HIScrollView should scroll its subview when the user clicks a scroll bar arrow. For example, this might be 10 pixels vertically and 20 pixels horizontally.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

`kEventParamOrigin`

A value of type `typeHIPoint` representing the origin. The origin is the current view-relative origin within the total scrollable image that is displayed at the top left corner of the view. These coordinates should always be greater than or equal to zero and less than or equal to the view's image size minus its view size. Typically, a view that implements the `kEventScrollableScrollTo` event by calling `HIViewSetBoundsOrigin` (page 91) returns the current bounds origin for this parameter, and a view implements the `ScrollTo` event by storing the origin in its instance data returns its stored origin for this parameter. For example, a scrollable view that currently is displaying page ten of a hundred page document would return a horizontal origin of zero and a vertical origin equal to ten times the height of the page.

Available in Mac OS X v10.2 and later.

Declared in `HIScrollView.h`.

Search Field Attribute Constants

Specify constants for search field attributes.

```
enum {
    kHISearchFieldNoAttributes = 0,
    kHISearchFieldAttributesCancel = (1 << 0),
    kHISearchFieldAttributesSearchIcon = (1 << 1)
};
```

Constants

`kHISearchFieldNoAttributes`

Indicates that this view does not have any attributes.

Available in Mac OS X v10.3 and later.

Declared in `HISearchField.h`.

`kHISearchFieldAttributesCancel`

Indicates that this view contains a Cancel button.

Available in Mac OS X v10.3 and later.

Declared in `HISearchField.h`.

`kHISearchFieldAttributesSearchIcon`

Indicates that this view contains the Search icon in the text field. If a menu is associated with the search field, this attribute is implicitly set and the Search icon will display with a menu disclosure badge.

Available in Mac OS X v10.4 and later.

Declared in `HISearchField.h`.

Discussion

These constants are used when calling `HISearchFieldCreate` (page 39), `HISearchFieldGetAttributes` (page 40), and `HISearchFieldChangeAttributes` (page 38).

Search Field Data Tags

Specify constants for search field data tags.

Discussion
Search field views support these tags previously defined for the `EditUnicodeText` control. These tags are available through `GetControlData` and `SetControlData` with a `ControlPartCode` of `kControlEditTextPart`:

- `kControlFontStyleTag`
- `kControlEditTextFixedTextTag`
- `kControlEditTextTextTag`
- `kControlEditTextKeyFilterTag`
- `kControlEditTextValidationProcTag`
- `kControlEditUnicodeTextPostUpdateProcTag`
- `kControlEditTextSelectionTag`
- `kControlEditTextKeyScriptBehaviorTag`
- `kControlEditTextCharCount`
- `kControlEditTextCFStringTag`

Availability
Available in Mac OS X v10.3 and later.

Search Field Part Code Constants

Specify constants for search field part codes.

```
enum {
    kControlSearchFieldCancelPart = 30,
    kControlSearchFieldMenuPart = 31
};
```

Constants

`kControlSearchFieldCancelPart`
Cancel part.
Available in Mac OS X v10.3 and later.
Declared in `HISearchField.h`.

`kControlSearchFieldMenuPart`
Menu part.
Available in Mac OS X v10.3 and later.
Declared in `HISearchField.h`.

Segment Attribute Constants

Specify segment attribute constants.

```
enum {
    kHISegmentNoAttributes = 0,
    kHISegmentSendCmdToUserFocus = (1 << 0)
};
```

Constants

`kHISegmentNoAttributes`

Indicates no attributes.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

`kHISegmentSendCmdToUserFocus`

If this attribute bit is set, the command that is sent when the segment is clicked will be directed at the user focus instead of up the segmented view's containment hierarchy.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

Discussion

These constants are used when calling `HISegmentedViewCreate` (page 44) and `HISegmentedViewChangeSegmentAttributes` (page 42).

Segment Behavior Constants

Specify segment behavior constants.

```
enum {
    kHISegmentBehaviorMomentary = 1,
    kHISegmentBehaviorRadio = 2,
    kHISegmentBehaviorToggles = 3,
    kHISegmentBehaviorSticky = 4
};
```

Constants

`kHISegmentBehaviorMomentary`

Pops back up after being pressed, just like a push button.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

`kHISegmentBehaviorRadio`

Stays pressed until another segment with the radio behavior is pressed. This makes the segment behave like a radio button. After this segment is clicked, the segmented view's value is changed to this segment's one-based index.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

`kHISegmentBehaviorToggles`

Behaves like a check box. When clicked, it toggles back and forth between checked and unchecked states. Currently, this constant should not be used; if you use it, you get the same effect as if you used `kHISegmentBehaviorMomentary`.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

`kHISegmentBehaviorSticky`
After being pressed, this type of segment stays pressed until it is unpressed programmatically. Currently, this constant should not be used; if you use it, you get the same effect as if you used `kHISegmentBehaviorMomentary`.

Available in Mac OS X v10.3 and later.

Declared in `HISegmentedView.h`.

Discussion
These constants are used in conjunction with `HISegmentedViewSetSegmentBehavior` (page 49) and `HISegmentedViewGetSegmentBehavior` (page 45).

Standard View Constants

Specify IDs of views that are commonly used.

```
const HIViewID kHIViewWindowContentID;
const HIViewID kHIViewWindowGrowBoxID;
const HIViewID kHIViewMenuContentID;
```

Constants
`kHIViewWindowContentID`
The standard view ID for the content view of a window.

Available in Mac OS X v10.2 and later.

Declared in `HIWindowViews.h`.

`kHIViewWindowGrowBoxID`
The standard view ID for the grow box view of a window. Not all windows have grow boxes, so you might not find this view if you look for it.

Available in Mac OS X v10.2 and later.

Declared in `HIWindowViews.h`.

`kHIViewMenuContentID`
The standard view ID for the content view of a menu. The Menu Manager assigns this view ID to all menu content views.

Available in Mac OS X v10.3 and later.

Declared in `HIMenuView.h`.

Text Field Event Constants

Specify constants for text field events (`kEventClassTextField`).

```
enum {
    kEventTextAccepted = 1,
    kEventTextShouldChangeInRange = 2,
    kEventTextDidChange = 3
};
```

Constants

`kEventTextAccepted`

This event is sent as a notification when the text contained in a control's editable text field has been accepted by the user. Text is accepted when the user presses Return or Enter on the keyboard for the `EditUnicodeText`, `HIComboBox`, and `HISearchField` controls, or when the text has changed in the field and the field loses focus for the `EditUnicodeText`, `HIComboBox`, `HISearchField` and `HITextView` controls. This event is sent to the control containing the text field only, it is not propagated. It is sent to all handlers installed on the control containing the text field.

Available in Mac OS X v10.3 and later.

Declared in `HITextViews.h`.

`kEventTextShouldChangeInRange`

This event is sent whenever the text is about to be modified in text field, either by user input or in other scenarios such as a paste from the clipboard, spell-checking word correction, or Mac OS X Service operation. You can change the text that is inserted by providing a replacement string as a parameter to this event. This event is only sent for Unicode text controls; it is not sent for the Classic non-Unicode `EditText` control. This event is not sent prior to programmatic modification of the text field contents using `SetControlData`. This event is not sent while an active inline editing session is in progress. Once the inline text has been confirmed, this event is sent prior to the confirmed text being inserted into the text field. If you need control over keystrokes during an inline editing session, you can use the `kEventTextInputFilterText` event. This event is sent to the control containing the text field only; it does not propagate.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

`kEventTextDidChange`

This event is sent to indicate that the contents of an editable text field have changed. This event is sent by all of the Unicode-based editable text views: `HIComboBox`, `HISearchField`, `HITextView` and `EditUnicodeText`. This event is not sent for the Classic non-Unicode `EditText` control. Note that this event is sent after inline editing operations, such as pressing a dead key, or using a input method that creates an inline editing hole. Most clients of this event should ignore the event during inline editing, and only respond to changes to the text after inline editing completes. A client can check for the presence of the `kEventParamUnconfirmedRange` parameter to determine whether inline editing is currently active; if this parameter is present, the client may wish to ignore the event. This event is not sent after programmatic modification of the text field contents using `SetControlData`. This event is sent only to the control containing the text field; it does not propagate. It is sent to all handlers registered for it.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

Discussion

Table 5 shows the event parameters associated with these events.

Table 5 Parameter names and types for text field events

Event kind	Parameter name	Parameter type
`kEventTextAccepted`	`kEventParamDirectObject`	`typeControlRef`

kEventTextShouldChangeInRange	kEventParamTextSelection	typeCFRange
	kEventParamCandidateText	typeCFStringRef
	kEventParamReplacementText (Optional)	typeCFStringRef
kEventTextDidChange	kEventParamUnconfirmedRange	typeCFRange
	kEventParamUnconfirmedText	typeCFStringRef

Text Field Event Parameter Constants

Specify constants for text field event parameters.

```
enum {
    kEventParamTextSelection = 'txsl',
    kEventParamCandidateText = 'tstx',
    kEventParamReplacementText = 'trtx',
    kEventParamUnconfirmedRange = 'tunr',
    kEventParamUnconfirmedText = 'txun'
};
```

Constants

`kEventParamTextSelection`

The range of the selection that is about to be changed. The units of the selection are in the same units that are returned in a `EditTextSelectionRec`, when called with `GetControlData` using `kControlEditTextSelectionTag`.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

`kEventParamCandidateText`

The text that is to replace the selection. Note that this string was originally created with `CFStringCreateWithCharactersNoCopy`, and the original text has a limited life span. If for some reason you need to retain the text past the end of your event handler, you should extract the characters from the string with `CFStringGetCharacters`, and then store those characters or create a new `CFString` from them.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

`kEventParamReplacementText`

Optional replacement text.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

`kEventParamUnconfirmedRange`

If the text field currently has an open inline hole, this parameter contains the range of text inside the hole. This parameter is optional and is only present during inline editing.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

`kEventParamUnconfirmedText`
If the text field currently has an open inline hole, this parameter contains the non-confirmed text currently being edited inside the hole. This parameter is optional and is only present during inline editing. Note that this string was originally created with `CFStringCreateWithCharactersNoCopy`, and the original text has a limited life span. If for some reason you need to retain the text past the end of your event handler, you should extract the characters from the string with `CFStringGetCharacters`, and then store those characters or create a new `CFString` from them.

Available in Mac OS X v10.4 and later.

Declared in `HITextViews.h`.

Transformation Constants

Specify transformation constants.

```
enum {
    kHITransformNone = 0x00,
    kHITransformDisabled = 0x01,
    kHITransformSelected = 0x4000
};
```

Constants

`kHITransformNone`
No visual transformation should be applied.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHITransformDisabled`
The image should be transformed to use a disabled appearance. This transformation should not be combined with any other transformations.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

`kHITransformSelected`
The image should be transformed to use a selected appearance. This transformation should not be combined with any other transformations.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

Discussion
These constants are used when calling `HICreateTransformedCGImage` (page 27).

kHIViewKindSignatureApple

Specify the signature of all HIToolbox views.

```
enum {
    kHIViewKindSignatureApple = 'appl'
};
```

Constants

`kHIViewKindSignatureApple`

Signature of all toolbox views.

Available in Mac OS X v10.4 and later.

Declared in `HIView.h`.

Result Codes

This table lists the result codes defined in HIView.

Result Code	Value	Description
`errNeedsCompositedWindow`	-30598	This result code is returned by an HIView or a Control Manager function when an action that requires a compositing window is attempted on a non-compositing window. It may also be returned when the value of a parameter is not valid for the requested action, even though the window is a compositing window. Available in Mac OS X v10.3 and later.

Document Revision History

This table describes the changes to *HIView Reference*.

Date	Notes
2007-04-13	Made minor format and editorial changes.
	Added information about the functions `HICocoaViewCreate` (page 20), `HICocoaViewSetView` (page 21), and `HICocoaViewGetView` (page 21).
2005-09-08	Updated for Mac OS X v10.4. Fixed bugs.
2003-05-15	New document that describes the C API for creating and manipulating view-based controls and menus in Carbon applications.

Document Revision History

Index

C

E

H

K

M

S

T