Ink Services Reference

Carbon > Events & Other Input

2006-01-10

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

Simultaneously published in the United States and Canada.

Contents

Tables

Ink Services Reference

Framework: Carbon/Carbon.h

Declared in Ink.h

Overview

Ink is a technology that allows users to enter text by writing with a stylus on a graphics tablet without requiring any modifications to the application that receives the text. As text is written on a tablet, it is automatically recognized and entered as a stream of key-down events into a document or text field.

The Ink Services application programming interface provides a set of functions that enables you to customize Ink input for your application. Using the Ink Services API, you can:

- Programmatically turn handwriting recognition on or off for your application
- Access Ink data at multiple levels (as points and recognized text)
- Support gestures that allow the user to manipulate text directly
- Set up deferred recognition or use on-demand recognition
- Access alternate text interpretations
- Manage options for drawing Ink
- Create and terminate an Ink phrase

Ink Services provides Ink input data (text interpretations, gestures, and so forth) through the Carbon Event Manager. Your application must set up one or more handlers to receive Ink-related events and to extract the relevant parameters from the events of interest to your application.

Functions by Task

Customizing Ink Services

`InkSetApplicationWritingMode` (page 11)
Controls where the user is allowed to write in the current application.

`InkSetApplicationRecognitionMode` (page 10)
Specifies whether Ink input should be interpreted as text, gestures, both, or neither.

`InkSetPhraseTerminationMode` (page 12)
Sets the conditions that define a phrase termination.

`InkSetDrawingMode` (page 12)
Controls what is drawn when the user writes.

Obtaining Information About Ink Services

`InkUserWritingMode` (page 24)
Returns the Ink writing mode set by the user in the Ink preferences pane.

`InkIsPhraseInProgress` (page 10)
Returns whether Ink Services has initiated and is currently maintaining an Ink phrase whose source is user input.

Handling Ink Phrases

`InkAddStrokeToCurrentPhrase` (page 9)
Adds a stroke to the current Ink phrase.

`InkTerminateCurrentPhrase` (page 15)
Terminates the current phrase.

Working With Alternate Text Interpretations

`InkTextAlternatesCount` (page 16)
Returns the number of alternate text interpretations available for an Ink phrase.

`InkTextCreateCFString` (page 17)
Obtains the string associated with a text interpretation of an Ink phrase.

`InkTextInsertAlternatesInMenu` (page 22)
Inserts a list of alternate text interpretations into a menu.

Working With Ink Text Objects

`InkTextKeyModifiers` (page 24)
Returns a value that specifies the key modifiers applied to an Ink phrase.

`InkTextCopy` (page 17)
Copies an existing Ink text object.

`InkTextBounds` (page 16)
Returns the bounds of an Ink text object.

`InkTextDraw` (page 18)
Rescales and draws Ink text into the specified bounds.

`InkTextGetTypeID` (page 22)
Returns the `CFTypeID` of an `InkTextRef` object.

Flattening and Unflattening Ink Text Objects

`InkTextFlatten` (page 19)
Flattens an Ink text object for archiving.

`InkTextCreateFromCFData` (page 18)
Creates an Ink text object from a previously-flattened Ink text object.

Working with Ink Stroke Objects

`InkTextGetStroke` (page 20)
Returns a reference to the specified stroke in an `InkTextRef`.

`InkTextGetStrokeCount` (page 21)
Returns the number of strokes in the specified `InkTextRef`.

`InkStrokeGetPointCount` (page 13)
Returns the number of points in the specified `InkStrokeRef`.

`InkStrokeGetPoints` (page 14)
Fills an array with the points belonging to the specified `InkStrokeRef`.

`InkStrokeGetTypeID` (page 15)
Returns the `CFTypeID` of an `InkStrokeRef` object.

Functions

InkAddStrokeToCurrentPhrase

Adds a stroke to the current Ink phrase.

```
void InkAddStrokeToCurrentPhrase (
   unsigned long iPointCount,
   InkPoint *iPointArray
);
```

Parameters

iPointCount
The number of elements in the `iPointArray` array.

iPointArray
A pointer to an array of `InkPoint` structures that specify the path of the stylus, starting with the point that defines the first stylus-down location and ending with the point that defines the last stylus-down location.

Discussion
This function operates on the Ink source from the application, and not on that from direct user input. So there is no need to specify the Ink source as `kInkSourceApplication`. See "Ink Source Types" (page 34) for more information on sources.

You do not need to call this function unless you have raw data to process or you have turned off automatic recognition (by calling the function `InkSetApplicationWritingMode` (page 11)) and have set up your application to handle Ink input events itself. For example, you might need to handle Ink input if your application needs to acquire pen data in a device-specific manner.

If your application handles Ink input events, it can still take advantage of the recognition service provided by Ink Services. To do so, your application should call the function `InkAddStrokeToCurrentPhase` to add one stroke at a time to the current phrase. You then terminate the phrase at the appropriate time by calling the function `InkTerminateCurrentPhrase` (page 15). Note that calling `InkAddStrokeToCurrentPhase` adds a stroke to the current phrase, but does not draw the stroke. See *Using Ink Services in Your Application* for details on writing code that uses the function `InkAddStrokeToCurrentPhase` to implement deferred recognition.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkIsPhraseInProgress

Returns whether Ink Services has initiated and is currently maintaining an Ink phrase whose source is user input.

```
Boolean InkIsPhraseInProgress (
   void
);
```

Return Value
Returns `TRUE` if the user is currently engaged in Ink input; `FALSE` otherwise.

Discussion
If your application manages its own phrase termination, you should use this function to make sure there is a phrase that can be terminated before you call the function `InkTerminateCurrentPhrase`. Don't call this function if the Ink data stream originates from your application rather than directly from user input. The application data stream is completely independent of the user data stream. If your application builds its own Ink phrases by calling the function `InkAddStrokeToCurrentPhrase`, it should be able track whether such a phrase is in-progress or not.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkSetApplicationRecognitionMode

Specifies whether Ink input should be interpreted as text, gestures, both, or neither.

```
void InkSetApplicationRecognitionMode (
   InkRecognitionType iRecognitionType
);
```

Parameters

iRecognitionType

The recognition mode you want Ink Services to use. Pass `kInkRecognitionGesture` to specify gesture recognition, `kInkRecognitionText` to specify text recognition, `kInkRecognitionNone` to turn off recognition, or `kInkRecognitionDefault` (which is `kInkRecognitionGesture | kInkRecognitionText`) to specify both gesture and text recognition. See "Recognition Modes" (page 30) for more information on the constants you can supply.

Discussion

This function only affects recognition of Ink that originates from the user. It does not affect recognition of Ink that originates from your application, and is recognized using the function `InkAddStrokeToCurrentPhrase`. Note that only text recognition (not gesture recognition) is performed on an Ink data stream that originates from your application.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.3 and later.

Declared In

`Ink.h`

InkSetApplicationWritingMode

Controls where the user is allowed to write in the current application.

```
void InkSetApplicationWritingMode (
   InkApplicationWritingModeType iWriteWhere
);
```

Parameters

iWriteWhere

An "Application Modes" (page 28) constant that specifies the Ink writing mode to use for your application. Pass `kInkWriteAnywhereInApp` if you want your application to allow Ink input and recognition and to receive Ink events when the user writing mode is set to `kInkWriteInkAwareOnly`. When you call this function with the `iWriteWhere` parameter set to `kInkWriteAnywhereInApp`, your application can receive Ink events whose screen locations lie outside the application windows. Pass `kInkWriteNowhereInApp` to disable Ink input temporarily, such as when the user is using a paint tool.

Discussion

You can call the function `InkSetApplicationWritingMode` to control when Ink input and recognition are allowed in your application. Using this function, you can turn Ink Services on or off for your application. Note that Ink input is available for your application only when your application is frontmost and when the user has turned on recognition in the Ink preferences pane.

If your application calls the function `InkSetApplicationWritingMode` with the parameter `kInkWriteNowhereInApp` to disable Ink Services management of pen events because you want to accumulate Ink data yourself, be aware that you may need to manage mouse event coalescing yourself. You can use the Carbon Event Manger function `SetMouseCoalescingEnabled` for this purpose. See *Using Ink Services in Your Application* for a discussion of mouse coalescing.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkSetDrawingMode

Controls what is drawn when the user writes.

```
void InkSetDrawingMode (
   InkDrawingModeType iDrawingMode
);
```

Parameters

`iDrawingMode`

A "Drawing Modes" (page 28) constant that specifies the drawing mode to use for your application. The default (`kInkDrawInkAndWritingGuides`) is for Ink Services to draw both the Ink writing guides and the Ink. Pass `kInkDrawInkOnly` if you want Ink Services to draw only the Ink. Pass `kInkDrawNothing` to turn off drawing of both the Ink writing guides and the Ink.

Discussion
Normally Ink Services draws writing guides, similar in look to the alternating solid and broken lines used on many paper writing tablets. The Ink itself is drawn anti-aliased and grayscale. Your application can call the function `InkSetDrawingMode` to request that Ink Services not draw the writing guides or not draw either Ink or the writing guides. If Ink drawing is disabled, your application must receive the points (by installing a handler for `kEventInkPoint` events) and draw the Ink.

You do not need to call the function `InkSetDrawingMode` to inhibit drawing if you called the function `InkSetApplicationWritingMode`, passing the value `kInkWriteNowhereInApp`. Also, Ink Services will not draw any point for which a `kEventInkPoint` Carbon event handler returns `noErr`.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkSetPhraseTerminationMode

Sets the conditions that define a phrase termination.

```
void InkSetPhraseTerminationMode (
   InkSourceType iSource,
   InkTerminationType iAllowedTerminationTypes
);
```

Parameters

iSource

An "Ink Source Types" (page 34) constant that specifies the source of the Ink data stream. You can use one of these constants to get independent control over termination of data originating with the user versus data that is passed from your application to Ink Services. To manage phrase termination for user input, pass the constant `kInkSourceUser`. To manage phrase termination for application input (that is recognized using the function `InkAddStrokeToCurrentPhrase`, pass the constant `kInkSourceApplication`.

iAllowedTerminationTypes

A constant that specifies the conditions which define a phrase termination. To turn off automatic phrase termination, pass `kInkTerminationNone`. You can restore the default phrase termination behavior by passing the constant `kInkTerminationDefault`. See "Phrase Termination Modes" (page 29) for more information on the constants you can supply.

Discussion

The default behavior is for Ink Services to terminate a phrase when one of the following events occur:

- The user removes the stylus from the proximity of the tablet
- A specified period of time elapses in which the stylus is not pressed to the tablet (The user can control the period of time in the Ink preferences pane.)
- The user writes sufficiently far away from the previous Ink—either horizontally, or on a new line

You can use the function `InkSetPhraseTerminationMode` if your application does not want the default behavior or wants complete control over when Ink phrases are terminated. If you turn off automatic phrase termination, you must make sure you manage phrase termination appropriately for your application.

For example, if you want to force Ink drawn in a specific input window to be treated as a single phrase until the user presses a "finished-writing" button, you would call `InkSetPhraseTerminationMode` with the parameter `kInkTerminationNone` to turn off automatic phrase termination. Then you would need to install a Carbon event handler for the event `kEventInkPoint`. Your handler would examine the `kEventInkPoint` events, notice when a pen-down event occurs on the "finished-writing" button, and then terminate the phrase by calling the function `InkTerminateCurrentPhrase`. See *Using Ink Services in Your Application* for details on writing code to handle phrase termination.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.3 and later.

Declared In

`Ink.h`

InkStrokeGetPointCount

Returns the number of points in the specified `InkStrokeRef`.

```
CFIndex InkStrokeGetPointCount (
   InkStrokeRef iStrokeRef
);
```

Parameters

iStrokeRef

The `InkStrokeRef` to get the point count from.

Return Value

A `CFIndex` indicating the number of points contained in the specified `InkStrokeRef`.

Discussion

Given an `InkStrokeRef`, this function returns the number of points that stroke contains. Use this function to calculate the appropriate size of the buffer passed to `InkStrokeGetPoints` (page 14).

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.4 and later.

Declared In

`Ink.h`

InkStrokeGetPoints

Fills an array with the points belonging to the specified `InkStrokeRef`.

```
InkPoint * InkStrokeGetPoints (
   InkStrokeRef iStrokeRef,
   InkPoint *oPointBuffer
);
```

Parameters

iStrokeRef

The `InkStrokeRef` to get the points from.

oPointBuffer

The buffer into which the point data is to be copied.

Return Value

A pointer to the copied array of point data from the specified `InkStrokeRef`; this value is the same as the `oPointBuffer` address provided by the application.

Discussion

Given an `InkStrokeRef` and a point buffer, this function fills that buffer with the points belonging to that stroke.

The size of the point buffer must be at least the size of `InkStrokeGetPointCount( iStrokeRef ) * sizeof( InkPoint )`. For details, see `InkStrokeGetPointCount` (page 13). The pointer to the block of memory containing the ink points is returned as the result.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.4 and later.

Declared In

`Ink.h`

InkStrokeGetTypeID

Returns the `CFTypeID` of an `InkStrokeRef` object.

```
CFTypeID InkStrokeGetTypeID (
   void
);
```

Return Value
The `CFTypeID` of an `InkStrokeRef` object.

Discussion
Given an `InkStrokeRef`, this function returns its `CFTypeID`.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.4 and later.

Declared In
`Ink.h`

InkTerminateCurrentPhrase

Terminates the current phrase.

```
void InkTerminateCurrentPhrase (
   InkSourceType iSource
);
```

Parameters

iSource

An "Ink Source Types" (page 34) constant that specifies the source of the Ink data stream.To terminate a phrase that originates from application input (that is recognized using the function `InkAddStrokeToCurrentPhrase)`, pass the constant `kInkSourceApplication`.

If you are managing phrase termination that originates from direct user input, you can pass the constant `kInkSourceUser`. Note that this function is normally not used in this fashion, as most applications can let Ink Services terminate such phrases automatically.

Discussion
You do not need to call this function unless you have turned off automatic phrase termination (by calling the function `InkSetPhraseTerminationMode` (page 12)) and have set up your application to manage phrase termination. When you call the function `InkTerminateCurrentPhrase`, any Ink drawn by Ink Services is erased. If your application handles phrase termination, it can still take advantage of the recognition service provided by Ink Services.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkTextAlternatesCount

Returns the number of alternate text interpretations available for an Ink phrase.

```
CFIndex InkTextAlternatesCount (
   InkTextRef iTextRef
);
```

Parameters

iTextRef

On input, a reference to the Ink text object that specifies the Ink word whose alternate count you want to obtain. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParamInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

Return Value

Returns the number of interpretations available for the specified Ink phrase.

Discussion

You can obtain the string associated with a text interpretation by calling the function `InkTextCreateCFString` (page 17). If you want to display a list of the alternate text interpretations to the user, call the function `InkTextInsertAlternatesInMenu` (page 22).

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.3 and later.

Declared In

`Ink.h`

InkTextBounds

Returns the bounds of an Ink text object.

```
HIRect InkTextBounds (
   InkTextRef iTextRef
);
```

Parameters

iTextRef

On input, a reference to the Ink text object whose bounds you want to obtain. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParameterInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

Return Value

An `HIRect` data structure that defines the bounds of the specified Ink text object.

Discussion

The bounds are initially global coordinates, and may extend beyond your application's windows.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkTextCopy

Copies an existing Ink text object.

```
InkTextRef InkTextCopy (
   InkTextRef iTextRef
);
```

Parameters

iTextRef

On input, a reference to the Ink text object you want to copy. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParameterInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

Return Value
A reference to the newly-created Ink text object.

Discussion
You can use this function to implement copy-and-paste functions in a deferred-recognition application, or in a text application to retain Ink when text is copied and pasted. The retention count of the new `InkTextRef` is 1.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkTextCreateCFString

Obtains the string associated with a text interpretation of an Ink phrase.

```
CFStringRef InkTextCreateCFString (
   InkTextRef iTextRef,
   CFIndex iAlternateIndex
);
```

Parameters

iTextRef

On input, a reference to the Ink text object that specifies the Ink word for which you want to create a string. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParamInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

iIndex

The index that specfies the text interpretation for which you want to obtain a `CFString`. Text interpretations are stored in an array in ranked order, with the most-likely interpretation at index zero.

Return Value
A `CFStringRef` that specifies an interpretation for the given Ink text phrase. Returns `NULL` if the index you provide is invalid. Your application is responsible for releasing the returned `CFStringRef`.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkTextCreateFromCFData

Creates an Ink text object from a previously-flattened Ink text object.

```
InkTextRef InkTextCreateFromCFData (
   CFDataRef iFlattenedInkText,
   CFIndex iIndex
);
```

Parameters

`iFlattenedInkText`
On input, a reference to a `CFData` data structure that contains data from a previously-flattened Ink text object.

`iIndex`
The index at which to start reading the data.

Return Value
Returns a reference to the newly-created Ink text object. The retention count of the newly-created Ink text object is 1.

Discussion
You can unflatten an Ink text object that was previously flattened using the function `InkTextFlatten`. If you flattened more than one Ink text object to the `CFMutableData` data type, then you must call the function `InkTextCreateFromCFData` for each Ink text object you want to unflatten, specifying the index that defines the start of the Ink text data you want to unflatten.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkTextDraw

Rescales and draws Ink text into the specified bounds.

```
void InkTextDraw (
   InkTextRef iTextRef,
   CGContextRef iContext,
   const CGRect *iBounds,
   InkTextDrawFlagsType iFlags
);
```

Parameters

iTextRef

On input, a reference to the Ink text object whose text you want to rescale and draw. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParameterInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

iContext

The `CGContext` into which you want to draw. Drawing is relative to the specified `CGContextRef`, and subject to the usual window and clipping constraints. Pass `NULL` if you want to draw to the canonical context of the current port.

iBounds

On input, a `CGRect` data structure that specifies the bounds into which you want the Ink text object to be drawn.

iFlags

A "Text Drawing Flags" (page 34) constant that specifies drawing settings. Pass `kInkTextDrawDefault` to use the default system settings when drawing, `kInkTextDrawIgnorePressure` if you do not want to use pressure sensitive gradients, and `kInkTextDrawHonorContext` t o use the current Quartz context settings.

Discussion

The function `InkTextDraw` is useful to applications that implement deferred recognition or searchable Ink. The original points and bounds of the Ink text object are scaled and offset to fit the specified bounds, so subsequent calls to the function `InkTextBounds` return the rescaled bounds.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.3 and later.

Declared In

`Ink.h`

InkTextFlatten

Flattens an Ink text object for archiving.

```
CFIndex InkTextFlatten (
   InkTextRef iTextRef,
   CFMutableDataRef ioDataRef,
   CFIndex iIndex
);
```

Parameters

iTextRef

On input, a reference to the Ink text object you want to flatten. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParameterInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

ioDataRef

On input, a reference to a `CFMutableData` data type. On output, refers to the flattened data. Your application is responsible for creating and releasing the `CFMutableDataRef`.

iIndex

The index at which you want the data to be written.

Return Value

Returns the number of bytes added to the `ioDataRef`. Returns `0` if the operation is unsuccessful or `iTextRef` is `NULL` or empty.

Discussion

`CFMutableData` objects are extensible, which means you can flatten more than one Ink text object into a `CFMutableData` object. You store data in a `CFMutableData` object by specifying the index at which data is to be stored. The function `InkTextFlatten` accepts the starting index as an input parameter, writes all the Ink text object data to the specified `CFMutableData` object, and returns the byte count for the amount of data that is actually stored.

To flatten an additional Ink text object into the same `CFMutableData` object, you must supply a value for the `iIndex` parameter that specifies the byte location at which to start writing the data for the additional Ink text object. You can calculate this value by summing the byte count returned by the previous call with the value of the `iIndex` parameter you provided in the previous call.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.3 and later.

Declared In

`Ink.h`

InkTextGetStroke

Returns a reference to the specified stroke in an `InkTextRef`.

```
InkStrokeRef InkTextGetStroke (
   InkTextRef iTextRef,
   CFIndex iStrokeIndex
);
```

Parameters

iTextRef

The `InkTextRef` to get the stroke from.

iStrokeIndex

The index of the stroke for which you want to get an `InkStrokeRef`.

Return Value

An `InkStrokeRef` for the specified stroke of the specified `InkTextRef`.

Discussion

Given an `InkTextRef` and a stroke index (between 0 and `InkTextGetStrokeCount( iTextRef ) - 1`), this function returns the `InkStrokeRef` corresponding to the specified stroke index. For details, see `InkTextGetStrokeCount` (page 21).

The returned `InkStrokeRef` is guaranteed to persist only for the life of the `InkTextRef` from which it was obtained. If you want to use the `InkStrokeRef` after the `InkTextRef` has been released, you must call the function `CFRetain` and pass the `InkStrokeRef` to it.

When any Ink object reference is obtained from a Carbon event, it is guaranteed to persist only for the life of the event handler. If you want to use the object at some later time, you must call the function `CFRetain` and pass the object to it.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.4 and later.

Declared In

`Ink.h`

InkTextGetStrokeCount

Returns the number of strokes in the specified `InkTextRef`.

```
CFIndex InkTextGetStrokeCount (
   InkTextRef iTextRef
);
```

Parameters

iTextRef

The `InkTextRef` to get the stroke count from.

Return Value

The number of stokes in the specified `InkTextRef`.

Discussion

Given an `InkTextRef`, this function returns the number of strokes the `InkTextRef` contains.

Availability

Not available in CarbonLib 1.x.

Available in Mac OS X v10.4 and later.

Declared In
`Ink.h`

InkTextGetTypeID

Returns the `CFTypeID` of an `InkTextRef` object.

```
CFTypeID InkTextGetTypeID (
   void
);
```

Return Value
The `CFTypeID` of an `InkStrokeRef` object.

Discussion
Given an `InkTextRef`, this function returns its `CFTypeID`.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.4 and later.

Declared In
`Ink.h`

InkTextInsertAlternatesInMenu

Inserts a list of alternate text interpretations into a menu.

```
ItemCount InkTextInsertAlternatesInMenu (
   InkTextRef iTextRef,
   MenuRef iMenuRef,
   MenuItemIndex iAfterItem
);
```

Parameters

`iTextRef`

On input, a reference to an Ink text object that specifies the Ink word for which you want to provide a list of alternate text interpretations. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParamInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

`iMenuRef`

A reference to the menu into which you want to insert the list of alternate text interpretations. Ink Services attaches menu event handlers to this menu, so you should use this `MenuRef` directly, rather than copy items from the menu reference to another menu.

`iAfterItem`

A value that specifies the menu item after which you want to insert the list of alternate text interpretations. If the specified menu item is `0`, the text alternates are inserted at the head of the menu. If the specified menu item is greater than or equal to the existing number of menu items, the text alternates are appended to the end of the menu. Remember that the first item in a menu item array is numbered `1`, not `0`.

Return Value
Returns the number of menu items added to the menu. Returns `0` if the operation is not successful.

Discussion
The function `InkTextInsertAlternatesInMenu` allows your application to insert a list of text interpretations for a given Ink text phrase into an existing contextual menu. You should handle a list of alternate text interpretations as a standard contextual menu using the Menu Manager function `ContextualMenuSelect`.

When a user selects an item from the list of alternates, the list of alternates maintained by Ink Services are reordered automatically. This means that if you call the function `InkTextCreateCFString` with the parameter `iIndex` set to `0`, you obtain the newly selected item.

Thus the user's choice persists in internal system data structures without requiring your application to call additional functions. However your application must update its own internal data structures appropriately.

You must rebuild the menu to reflect the user's choice. After the user makes a choice and then reopens the menu, you must make sure the newly-selected item shows up as the first item in the menu. The items in the menu should mirror the list of alternates maintained by Ink Services.

Upon return from the function `ContextualMenuSelect`, your application can determine if the user has made a selection by checking the value of the parameter `outUserSelectionType`. The value indicates the item that the user selected from the contextual menu. If there is a selection, your application can examine the `outMenuID` and `outMenuItem` parameters of the function `ContextualMenuSelect`, and use these values to obtain the alternate text interpretation by calling the Menu Manager function `CopyMenuItemTextAsCFString`.

Menu items for a set of alternates whose first letter is an alphabetical character always include an alternate whose first letter is the opposite lettercase. Menu items for a set of alternates whose first letter is a non-alphabetical character do not include a lettercase alternate.

When the menu items are reordered automatically, the text that was first in the list moves to the second or the third position, depending upon whether the first letter is alphabetical or non-alphabetical. For example, the following list of menu items:

crash, Crash, crush, crust, wrash

If the user chooses crush, the menu items are reordered as follows:

crush, Crush, crash, crust, wrash

Notice that the list of alternates is kept to a length of five. A lettercase alternate for crush is added to the menu while the uppercase alternate Crash is dropped.

For a non-alphabetic first character, however, such as a number, the original moves to the second position. So for the following menu:

1239, 1234, 1289, 1284

If the user chooses 1234, the menu becomes:

1234, 1239, 1289, 1284

If it is important for your application to maintain the original order of alternates, then it must use its own internal data structures to keep track of the original list.

See *Using Ink Services in Your Application* for details on writing code that uses the function `InkTextInsertAlternatesInMenu` to implement a correction model.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`Ink.h`

InkTextKeyModifiers

Returns a value that specifies the key modifiers applied to an Ink phrase.

```
UInt32 InkTextKeyModifiers (
   InkTextRef iTextRef
);
```

Parameters

`iTextRef`

On input, a reference to the Ink text object whose key modifiers you want to obtain. You must obtain an Ink text object reference (`InkTextRef`) through your application's Ink event handler. Your handler must take care of the Carbon event class `kEventClassInk` and the event kind `kEventInkText`. The event parameter `kEventParameterInkTextRef` that you obtain from this event kind is a reference to an Ink text object.

Return Value
Returns a value that indicates which modifier keys were down during input of the Ink phrase. This value is in the same form as that used by the Carbon Event Manager for the event parameter `kEventParamKeyModifiers`.

Discussion
Ink Services assigns keyboard modifier keys to a stroke if those keys are held down for more than 50% of the stroke's points. Ink Services assigns the modifier keys associated with a phrase's first stroke to the entire phrase. Ink Services assigns the modifier keys associated with a phrase to all of the text interpretations that are derived from an Ink phrase.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

InkUserWritingMode

Returns the Ink writing mode set by the user in the Ink preferences pane.

```
InkUserWritingModeType InkUserWritingMode (
   void
);
```

Return Value
A value that specifies the current user preferences settings for Ink: `kInkWriteInInkAwareAppsOnly`, `kInkWriteAnywhere`, or `kInkWriteNowhere`. In general, Ink services are not available if `kInkWriteNowhere` is returned (indicating the user has turned Ink off entirely). See "User Writing Modes" (page 27) for more information on each of these constants.

Discussion
User preferences for Ink are set by the user in the Ink pane of System Preferences. Your application can only read these values, not set them.

Availability
Not available in CarbonLib 1.x.
Available in Mac OS X v10.3 and later.

Declared In
`Ink.h`

Data Types

InkTextRef

Defines a data type for a reference to an opaque Ink text object.

```
typedef     struct OpaqueInkTextRef * InkTextRef;
```

Discussion
You must use the Core Foundation functions `CFRetain` and `CFRelease` to manage the retention and release of Ink text objects. When an Ink text reference is obtained from a Carbon event, it is guaranteed to persist only for the life of the event handler. If your application needs to use the Ink text object at some later time, you must call the function `CFRetain`, passing the object you want to retain as a parameter.

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

Declared In
`Ink.h`

InkStrokeRef

Defines a data type for a reference to an opaque Ink stroke object.

```
typedef     struct OpaqueInkStrokeRef * InkStrokeRef;
```

Discussion
You must use the Core Foundation functions `CFRetain` and `CFRelease` to manage the retention and release of Ink stroke objects. When an Ink stroke reference is obtained from a Carbon event, it is guaranteed to persist only for the life of the event handler. If your application needs to use the Ink stroke object at some later time, you must call the function `CFRetain`, passing the object you want to retain as a parameter.

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

Declared In
`Ink.h`

InkAlternateCount

Defines a data type that specifies the number of alternate text interpretations of an Ink phrase.

```
typedef unsigned long     InkAlternateCount;
```

Discussion
Values of `type InkAlternateCount` are returned by the function `InkTextAlternatesCount` (page 16) and passed as a parameter to the function `InkTextCreateCFString` (page 17).

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

Declared In
`Ink.h`

InkPoint

Contains data that describes an Ink point.

```
struct InkPoint {
    HIPoint             point;
    TabletPointRec      tabletPointData;
    UInt32              keyModifiers;
};
typedef struct InkPoint      InkPoint;
typedef InkPoint *      InkPointPtr;
```

Fields

`point`

Defines a point in floating-point coordinates. These values are generally in global coordinates, with full sub-pixel accuracy. This coordinate is what you obtain for a mouse event from the Carbon event parameter `kEventParamMouseLocation`, which also contains a `typeHIPoint` value.

`tabletPointData`

A tablet point structure that contains pressure, tilt, rotation, and coordinate (in tablet space) data for a pen. The pressure value is a measure of how hard the pen is being pressed, ranging from 0 to 65535. Some tablet manufacturers allow users to adjust pen sensitivity. In these cases, the zero value always corresponds to the threshold set by the user, and the pressure value is relative to that threshold. See *Carbon Event Manager Reference* for information on the `TabletPointRec` data type.

`keyModifiers`

A value that specifies the keyboard modifier key that is pressed when the point is sampled. This value is in the same form as that used by the Carbon Event Manager for the event parameter `kEventParamKeyModifiers`.

Discussion
An `InkPoint` data structure contains an essentially complete set of per-point data. Ink Services currently only requires the point's (x,y) coordinates and pressure to perform recognition and to draw the ink, but future recognition services may require other information from the `TabletPointRec`.

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

Declared In
`Ink.h`

Constants

User Writing Modes

Specify the Ink writing mode set by the user in the Ink pane of System Preferences.

```
enum {
    kInkWriteNowhere     = 'nowh',
    kInkWriteAnywhere = 'anyw',
    kInkWriteInInkAwareAppsOnly     = 'iapp'
};
typedef FourCharCode InkUserWritingModeType;
```

Constants

`kInkWriteNowhere`

Specifies the user has disabled Ink or that Ink Services are not available (for example, a tablet is not attached).

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkWriteAnywhere`

Specifies the user has enabled Ink to allow writing anywhere on the screen. Ink Services flows ink points and recognition results to the frontmost application. This is the default situation when the user enables Ink.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkWriteInInkAwareAppsOnly`

Specifies the user has enabled Ink only to allow writing in an application that has enabled Ink Services by calling the function `InkSetApplicationWritingMode` with the `kInkWriteAnywhereInApp` parameter.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Discussion
These constants are returned by the function `InkUserWritingMode` (page 24).

Application Modes

Specify an Ink input mode to use for an application.

```
enum {
    kInkWriteNowhereInApp     = 'nowa',
    kInkWriteAnywhereInApp     = 'anya'
};
typedef FourCharCode     InkApplicationModeType;
```

Constants

`kInkWriteNowhereInApp`

Specifies not to allow Ink input in your application.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkWriteAnywhereInApp`

Specifies to allow Ink input anywhere onscreen for your application.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Discussion

You can supply these constants as parameters to the function `InkSetApplicationWritingMode` (page 11). If the user has not enabled Ink or if there is not an Ink input device available, then calling `InkSetApplicationWritingMode` (page 11) with the parameter `kInkWriteAnywhereInApp` has no effect.

Drawing Modes

Specify what Ink Services should draw.

```
enum {
    kInkDrawNothing = 0,
    kInkDrawInkOnly = 1,
    kInkDrawInkAndWritingGuides= 2
};
typedef unsigned long InkDrawingModeType;
```

Constants

`kInkDrawNothing`

Specifies not to draw Ink or the writing guides.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkDrawInkOnly`

Specifies to draw Ink but not the writing guides.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkDrawInkAndWritingGuides`

Specifies to draw both the Ink and the writing guides. This is the default.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Discussion
You can pass these constants as parameters to the function `InkSetDrawingMode` (page 12).

Phrase Termination Modes

Defines the conditions under which an Ink phrase should be terminated.

```
enum InkTerminationType{
    kInkTerminationNone = 0,
    kInkTerminationTimeOut = 1,
    kInkTerminationOutOfProximity = 1 << 1,
    kInkTerminationRecognizerHorizontalBreak = 1 << 2,
    kInkTerminationRecognizerVerticalBreak = 1 << 3,
    kInkTerminationStroke = 1 << 4,
    kInkTerminationAll = (unsigned long) 0xFFFFFFFF,
    kInkTerminationDefault = 0x0F
};
typedef unsigned long         InkTerminationType;
```

Constants

`kInkTerminationNone`
Specifies to inhibit automatic phrase termination by Ink Services.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTerminationTimeOut`
Specifies to terminate a phrase when all of the following are true:

- The user stops writing and lifts the stylus
- The user keeps the stylus within the proximity range of the tablet
- The user does not resume writing within the period of time defined by the user in the Ink pane of System Preferences

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTerminationOutOfProximity`
Specifies to terminate a phrase when the user stops writing and lifts the stylus entirely out of the proximity range of the tablet. This is on by default. However, users can turn off proximity termination in the Ink pane of System Preferences if they find it interferes with their writing style.

If the user turns off proximity termination, your application can't turn it on even if you call the function `InkSetPhraseTerminationMode` (page 12) with the parameter `kInkTerminationOutOfProximity`.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTerminationRecognizerHorizontalBreak`
Specifies to terminate a phrase when the user leaves a large horizontal space between words (approximately two character widths or more).

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTerminationRecognizerVerticalBreak`
Specifies to terminate a phrase when the user finishes one line and begins writing on the next.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTerminationStroke`
Causes phrases to be terminated at the end of every stroke (whenever the pen is lifted from the tablet while writing). Only useful for single-stroke gesture input, not for text.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

`kInkTerminationAll`
Specifies to restore automatic phrase termination by Ink Services. In this case, Ink Services uses all of the termination modes (except `kInkTerminationNone`) described previously. Deprecated in Mac OS X v10.4. As of Mac OS X v10.4, this value is overridden to behave like `kInkTerminationDefault`.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTerminationDefault`
Restores default phrase termination matching the current user settings (`kInkTerminationTimeOut | kInkTerminationOutOfProximity | kInkTerminationRecognizerHorizontalBreak | kInkTerminationRecognizerVerticalBreak`). See also `kInkTerminationOutOfProximity`.

Declared in `Ink.h`.

Available in Mac OS X v10.4 and later.

Discussion
An Ink phrase (represented as an `InkTextRef` in your application) is typically a word in a Roman language. Ink Services uses phrases to determine when to erase onscreen Ink and initiate recognition. You can pass Ink phrase termination constants as arguments to the function `InkSetPhraseTerminationMode` (page 12). You can combine two or more constants to obtain precise control over phrase termination.

Recognition Modes

Specify how to interpret Ink input for an application.

```
enum InkRecognitionType{
    kInkRecognitionNone     = 0,
    kInkRecognitionText     = 1,
    kInkRecognitionGesture     = 1 << 1,
    kInkRecognitionDefault     = 3
};
typedef unsigned long     InkRecognitionType;
```

Constants
`kInkRecognitionNone`
Specifies to turn off Ink recognition.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkRecognitionText`
Specifies to allow interpretation of Ink input as text.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkRecognitionGesture`
Specifies to allow interpretation of Ink input as gestures.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkRecognitionDefault`
Specifies the default setting, which is to interpret Ink input as text or gestures.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Discussion
The recognition type constants are used as arguments for the function `InkSetApplicationRecognitionMode` (page 10). You can use these constants to specify that Ink Services interprets input as both text and gestures or as either type individually.

Editing Gestures

Define editing actions.

```
enum InkGestureKind {
    kInkGestureUndo     = 'undo',
    kInkGestureCut     = 'cut ',
    kInkGestureCopy     = 'copy',
    kInkGesturePaste     = 'past',
    kInkGestureClear     = 'cler',
    kInkGestureSelectAll     = 'sall',
    kInkGestureLeftSpace     = 'lspc',
    kInkGestureRightSpace     = 'rspc',
    kInkGestureTab     = 'tab ',
    kInkGestureLeftReturn     = 'lrtn',
    kInkGestureRightReturn     = 'rrtn',
    kInkGestureDelete     = 'del ',
    kInkGestureEscape     = 'esc ',
    kInkGestureJoin     = 'join'
    };
typedef FourCharCode     InkGestureKind;
```

Constants
`kInkGestureUndo`
Specifies to undo the last action.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureCut`
Specifies to cut.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureCopy`
Specifies to copy.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGesturePaste`
Specifies to paste.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureClear`
Specifies to clear.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureSelectAll`
Specifies to select all items in the area that has user focus.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureLeftSpace`
Specifies to insert a single space character. The “left” distinction indicates that the gesture is drawn with the long, horizontal tail is on the left side.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureRightSpace`
Specifies to insert a single space character. The “right” distinction indicates that the gesture is drawn with the long, horizontal tail is on the right side.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureTab`
Specifies to insert a tab character.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureLeftReturn`
Specifies to insert a return (new line) character. The “left” distinction indicates that the gesture is drawn with the small angle-bracket pointing to the left side.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureRightReturn`
Specifies to insert a return (new line) character. The “right” distinction indicates that the gesture is drawn with the small angle-bracket pointing to the right side.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureDelete`
Specifies to delete. This corresponds to pressing the Delete key.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureEscape`
This corresponds to pressing the Escape key.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkGestureJoin`
Specifies to join two words into a single word, eliding the space between them, and may be applied to editable objects other than text. The gesture is similar in shape to the letter "v". The joined words are the ones closest to the top-most points of the gesture. This is a tentative, always targeted, gesture, meaning that the system treats the associated Ink tentatively as a gesture until your application either confirms the Ink is indeed a gesture or returns `eventNotHandledErr`, informing the system the Ink is not a gesture.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Discussion
These constants are returned in the Carbon event parameter `kEventParamInkGestureKind`. The Carbon event class for this parameter is `kEventClassInk` and the event kind is `kEventInkGesture`. The constants define the complete set of gestures recognized by Ink Services. When a gesture event is received by your application, your application should determine the gesture kind and then take appropriate action. For more details, see *Using Ink Services in Your Application*.

Alternates Menu Command IDs

Specify the menu command IDs assigned to items inserted in the alternates menu.

```
enum {
    kInkAlternateCommand = 'inka',
    kInkSeparatorCommand         = 'inks',
    kInkDrawingCommand             = 'inkd'
};
```

Constants
`kInkAlternateCommand`
Specifies the menu command ID assigned to menu items inserted by the function `InkTextInsertAlternatesInMenu` (page 22). You can use this constant to determine which menu items in a menu are supplied by Ink Services.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkSeparatorCommand`
Specifies the menu command ID assigned to the separator item between the alternates and the Ink drawing.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkDrawingCommand`
Specifies the menu command ID assigned to the menu item containing the ink drawing.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Text Drawing Flags

Specify settings to use when drawing Ink text.

```
enum unsigned long InkTextDrawFlagsType{
kInkTextDrawDefault         = 0,
kInkTextDrawIgnorePressure     = 1,
kInkTextDrawHonorContext     = 1 << 1
};
```

Constants

`kInkTextDrawDefault`
Specifies to use the default system settings when drawing. By default, Ink is drawn with pressure sensitive gradients, and the Quartz context settings are overridden for line color and width.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTextDrawIgnorePressure`
Specifies not to use pressure sensitive gradients when drawing.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkTextDrawHonorContext`
Specifies to use the current Quartz context settings for line color and width.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Ink Source Types

Specify sources for an Ink data stream.

```
enum unsigned long InkSourceType{
    kInkSourceUser                    = 1,
    kInkSourceApplication                        = 2
};
```

Constants

`kInkSourceUser`
Specifies the Ink source from direct user input.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

`kInkSourceApplication`
Specifies the Ink source from the application.

Available in Mac OS X v10.3 and later.

Declared in `Ink.h`.

Discussion
You can use these constants to specify which data stream is currently being controlled by calls to the functions `InkTerminateCurrentPhrase` and `InkSetPhraseTerminationMode`. You can control phrase termination for both the user-input data stream (`kInkSourceUser`) and an application-input data stream (`kInkSourceApplication`) independently.

Ink Pen Constants

Specify ink pen constants.

```
enum {
    kInkPenTipButtonMask = NX_TABLET_BUTTON_PENTIPMASK + 0,
    kInkPenLowerSideButtonMask = NX_TABLET_BUTTON_PENLOWERSIDEMASK
+ 0,
    kInkPenUpperSideButtonMask = NX_TABLET_BUTTON_PENUPPERSIDEMASK
+ 0
};
```

Constants
`kInkPenTipButtonMask`
The writing or eraser tip.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

`kInkPenLowerSideButtonMask`
The lower pen barrel button.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

`kInkPenUpperSideButtonMask`
The upper pen barrel button.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

Discussion
Pens used with modern graphics tablets often have multiple barrel buttons that can be assigned special meaning by the tablet driver or by an application. In addition, the writing or eraser tip may be engaged at any given moment. By performing an AND operation of these contsants with that of a `buttons` member of a `TabletPointRec` in Carbon or the value returned by the `buttonMask` message sent to tablet events (or mouse events containing tablet data) in Cocoa, you can determine which (if any) pen tip or barrel buttons are currently being held down. These buttons and `buttonMask` data are only available for tablet-point events (not tablet-proximity events).

To ensure consistency between the values used by driver writers and the values used by applications, these constants are defined in terms of `NX_` constants from `IOKit/hidsystem/IOLLEvent.h`.

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

Ink Tablet Constants

Specify ink tablet constants.

```
enum {
    kInkTabletPointerUnknown = NX_TABLET_POINTER_UNKNOWN + 0,
    kInkTabletPointerPen = NX_TABLET_POINTER_PEN + 0,
    kInkTabletPointerCursor = NX_TABLET_POINTER_CURSOR + 0,
    kInkTabletPointerEraser = NX_TABLET_POINTER_ERASER + 0
};
```

Constants

`kInkTabletPointerUnknown`

The type of tablet pointer is unknown; having an unknown type of tablet pointer should not happen.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

`kInkTabletPointerPen`

The writing end of a stylus-like device.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

`kInkTabletPointerCursor`

Any puck-like device.

Available in Mac OS X v10.4 and later.

Declared in `Ink.h`.

`kInktabletPointerEraser`

The eraser end of a stylus-like device.

Discussion

Pens used with modern graphics tablets often have a writing tip and an eraser tip. Some tablets also support pucks in addition to, or instead of, stylus-like devices. By comparing these constants to the contents of the `pointerType` element of a `TabletProximityRec` in Carbon or to the value returned by the `pointerType` message to tablet events (or mouse events with tablet data in them) in Cocoa, you can determine what kind of pointer device and which tip of a stylus-like device is being used with a graphics tablet. These `pointerType` data are only available in tablet-proximity events (not tablet-point events).

To ensure consistency between the values used by driver writers and the values used by applications, these constants are defined in terms of `NX_` constants from `IOKit/hidsystem/IOLLEvent.h`.

Availability

Available in Mac OS X v10.4 and later.

Result Codes

There are no results codes specific to Ink Services. Rather than returning `OSStatus` values, functions return `NULL` or specific, predetermined invalid responses when you pass invalid parameters to them. Designing the API in this way allows you to chain function calls and write code that is more compact.

Ink-Related Carbon Events

Table A-1 lists the Carbon event kinds and event parameters associated with the Carbon event class `kEventClassInk`. These constants are part of the Carbon Event Manager. See *Handling Carbon Events* for more information on using the Carbon Event Manger. See *Using Ink Services in Your Application* for information on obtaining Ink-related Carbon events and the associated event parameters.

Table A-1 Event kinds and parameters for the event class kEventClassInk

Carbon event kind	Event parameters	Type
`kEventInkPoint`	`kEventParamEventRef`	`typeEventRef`
`kEventInkGesture`	`kEventParamInkGestureKind`	`typeUInt32`
	`kEventParamInkGestureBounds`	`typeHIRect`
	`kEventParamInkGestureHotspot`	`typeHIPoint`
`kEventInkText`	`kEventParamInkTextRef`	`typePtr`
	`kEventParamInkKeyboardShortcut`	`typeBoolean`

Descriptions for each event parameters are as follows:

- `kEventParamEventRef` A reference to the original mouse event that spawned this `kEventInkPoint` event.
- `kEventParamInkGestureKind` An "Editing Gestures" (page 31) constant. These constants specify editing actions.
- `kEventParamInkGestureBounds` The rectangle that defines the bounds of a gesture.
- `kEventParamInkGestureHotspot` The location to which a targeted gesture should apply.
- `kEventParamInkTextRef` A reference to an opaque Ink text object (`InkTextRef`).
- `kEventParamInkTextKeyboardShortcut` A Boolean value that indicates whether the Ink associated with an Ink text object (`InkTextRef`) is likely a keyboard equivalent. The value is `TRUE` if the Command or Control key is pressed and the top-choice alternate text is a single character. Checking for this parameter provides an easy way for you to determine if an `InkTextRef` is likely to be a keyboard shortcut instead of text. Otherwise, to determine whether the Ink text is a keyboard shortcut, you would need to extract the `kEventParamInkTextRef` parameter, retrieve the `CFStringRef` for the text, determine the length of the string, and then check for modifier keys. In most cases, you don't need to handle this event, and can immediately return `eventNotHandledErr`.

Table A-2 (page 38) lists the event parameters for the Carbon event kind `kEventAppIsEventInInstantMouser` which is of class `kEventClassApplication`. This event is sent to your application when the system needs to determine if the global mouse location of the given event coincides with an instant-mousing area. An **instant-mousing area** is an area where a mouse-down event should only be interpreted as a mousing action; the event should not generate Ink.

The instant-mousing event is dispatched only when a stylus is initially pressed to a tablet, at the beginning of a phrase, before Ink Services has determined whether the user is writing or not. Once the user has begun writing, stylus-down actions do not generate instant-mousing events. How your application responds to this event determines whether Ink Services treats the stylus-down action as Ink or not. The instant-mousing status of all standard Carbon and Cocoa controls is determined automatically. You need only install an instant-mouse event handler if your application defines custom controls you want to designate as instant mousing areas.

Table A-2 Event parameters and types for the event kind kEventAppIsEventInInstantMouser

Event parameters	Type
`kEventParamEventRef`	`typeEventRef`
`kEventParamIsInInstantMouser`	`typeBoolean`

Descriptions for each event parameter are as follows:

- `kEventParamEventRef` A reference to the original mouse event that spawned this `kEventParamIsInInstantMouser` event. This mouse event contains the point your application must evaluate to determine if the point is in an instant-mousing area.
- `kEventParamIsInInstantMouser` A Boolean value that specifies whether the point is in an instant-mousing area (`TRUE`) or not (`FALSE`). Your application must set this parameter to define instant-mousing screen regions.

Document Revision History

This table describes the changes to *Ink Services Reference*.

Date	Notes
2006-01-10	Updated for Mac OS X v10.4.
	Added descriptions of the `InkStrokeGetPointCount`, `InkStrokeGetPoints`, `InkStrokeGetTypeID`,`InkTextGetStroke`, `InkTextGetStrokeCount`, and `InkTextGetTypeID` functions.
	Added descriptions of these data types and constants: `InkStrokeRef`,`kInkPenLowerSideButtonMask`, `kInkPenTipButtonMask`,`kInkPenUpperSideButtonMask`, `kInkTabletPointerCursor`,`kInkTabletPointerEraser`, `kInkTabletPointerPen`, `kInkTabletPointerUnknown`,`kInkTerminationDefault`, and `kInkTerminationStroke`.
2003-07-24	Added the constants "Ink Source Types" (page 34), "Text Drawing Flags" (page 34).
	Added additional constants to "Alternates Menu Command IDs" (page 33).
	Added a parameter to and additional information about the usage of the functions "InkSetPhraseTerminationMode" (page 12), "InkTerminateCurrentPhrase" (page 15), and "InkTextDraw" (page 18).
	Added additional information on the usage of the functions "InkSetApplicationRecognitionMode" (page 10), "InkIsPhraseInProgress" (page 10), "InkAddStrokeToCurrentPhrase" (page 9).
	Changed the return type for the function "InkTextBounds" (page 16).
2003-06-19	First release of this document. This is a preliminary version.

Document Revision History

Index

P

R

T

U