# Drag Manager Reference

**Carbon > Interapplication Communication**

**2006-07-12**

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



Simultaneously published in the United States and Canada.

# Contents

# Drag Manager Reference

**Framework:** Carbon/Carbon.h

**Declared in** Drag.h

## Overview

The Drag Manager facilitates dragging objects within the Macintosh user interface. The Drag Manager provides functions that handle the user interface for dragging an object from, within, or to one of your application's windows. The Drag Manager can be used whenever an object is dragged within your application.

Use the Drag Manager if you want your users to be able to drag items within your own application's windows or between those of your application and other applications. You can also use the Drag Manager to allow the user of your application to drag selections of your documents to the Finder to create "clippings" from your documents and to allow selections from other applications to be dragged directly into your documents.

## Functions by Task

### Installing and Removing Drag Handlers

`InstallReceiveHandler` (page 62) Deprecated in Mac OS X v10.5
Installs a receive handler function for one or all of your application's windows.

`InstallTrackingHandler` (page 62) Deprecated in Mac OS X v10.5
Installs a tracking handler function for one or all of your application's windows.

`RemoveReceiveHandler` (page 67) Deprecated in Mac OS X v10.5
Removes a receive handler function from one or all of your application's windows.

`RemoveTrackingHandler` (page 67) Deprecated in Mac OS X v10.5
Removes a tracking handler function from one or all of your application's windows.

### Creating and Disposing of Drag References

`NewDrag` (page 17)
Creates a new drag reference for your application to use with the Drag Manager.

`DisposeDrag` (page 12)
Disposes of a drag reference and its associated data.

## Adding Drag Item Flavors

`AddDragItemFlavor` (page 50) Deprecated in Mac OS X v10.5
Adds a flavor to a drag item, creating a new item if necessary.

`SetDragItemFlavorData` (page 69) Deprecated in Mac OS X v10.5
Sets the data or part of the data contained within an existing flavor.

## Providing Drag Callback Functions

`SetDragInputProc` (page 20)
Sets the drag input function for the Drag Manager to use with a particular drag.

`SetDragDrawingProc` (page 68) Deprecated in Mac OS X v10.5
Sets the drag drawing function for the Drag Manager to use with a particular drag.

`SetDragSendProc` (page 70) Deprecated in Mac OS X v10.5
Sets the send data function for the Drag Manager to use with a particular drag.

## Setting the Drag Image

`SetDragImageWithCGImage` (page 19)
Associates a Core Graphics image with a drag reference.

`SetDragImage` (page 49) Deprecated in Mac OS X v10.4
Associates an image with a drag reference. (Deprecated. Use `SetDragImageWithCGImage` (page 19) instead.)

## Altering the Behavior of a Drag

`ChangeDragBehaviors` (page 11)
Changes the behavior of a drag.

## Performing a Drag

`TrackDrag` (page 22)
Drags an item or collection of items from your application.

## Getting Drag Item Information

`GetDragItemBounds` (page 14)
Gets the bounding rectangle of a drag item.

`SetDragItemBounds` (page 21)
Sets the bounding rectangle of a drag item.

`CountDragItemFlavors` (page 51) Deprecated in Mac OS X v10.5
Gets the number of flavors that are contained within a drag item.

`CountDragItems` (page 52) Deprecated in Mac OS X v10.5
Gets the number of drag items that are contained in a drag reference.

`GetDragItemReferenceNumber` (page 56) Deprecated in Mac OS X v10.5
Gets the reference number of a specific item in a drag reference.

`GetFlavorData` (page 57) Deprecated in Mac OS X v10.5
Gets all or part of the data for a specific flavor in a drag item.

`GetFlavorDataSize` (page 58) Deprecated in Mac OS X v10.5
Gets the size of the data for a specific flavor in a drag item.

`GetFlavorFlags` (page 59) Deprecated in Mac OS X v10.5
Gets the flags for a specific flavor in a drag item.

`GetFlavorType` (page 60) Deprecated in Mac OS X v10.5
Gets the type of a specific flavor in a drag item.

## Getting and Setting the Drop Location

`GetDropLocation` (page 57) Deprecated in Mac OS X v10.5
Gets the Apple Event descriptor of the drop location.

`GetStandardDropLocation` (page 60) Deprecated in Mac OS X v10.5
Gets the standard drop location set by the receiver of a drag.

`SetDropLocation` (page 71) Deprecated in Mac OS X v10.5
Sets the Apple Event descriptor for the drop location of a drag.

`SetStandardDropLocation` (page 71) Deprecated in Mac OS X v10.5
Used by the receiver of a drag to set the standard drop location for a drag.

## Getting Drag Status Information

`GetDragAttributes` (page 13)
Gets the current set of drag attribute flags.

`GetDragMouse` (page 15)
Gets the current mouse and pinned mouse locations.

`SetDragMouse` (page 21)
Sets the current pinned mouse location.

`GetDragOrigin` (page 16)
Gets the `mouseDown` parameter location that started the given drag.

`GetDragModifiers` (page 15)
Gets the current set of keyboard modifiers.

## Accessing Drag Actions

`GetDragAllowableActions` (page 12)
Returns the actions that the drag receiver may take on the data within a drag.

`SetDragAllowableActions` (page 18)
Sets the actions that the drag receiver may take on the data within a drag.

`GetDragDropAction` (page 14)
Returns the action performed by the receiver of the drag.

`SetDragDropAction` (page 19)
Sets the action performed by the receiver of the drag.

## Highlighting a Drag

`DragPostScroll` (page 54) Deprecated in Mac OS X v10.5
Restores the drag highlight after scrolling part of your window.

`DragPreScroll` (page 54) Deprecated in Mac OS X v10.5
Prepares your window or pane for scrolling.

`GetDragHiliteColor` (page 55) Deprecated in Mac OS X v10.5
Returns the drag highlight color for a window.

`HideDragHilite` (page 61) Deprecated in Mac OS X v10.5
Removes highlighting created with the `ShowDragHilite` function.

`ShowDragHilite` (page 72) Deprecated in Mac OS X v10.5
Highlights an area of your window during a drag.

`UpdateDragHilite` (page 73) Deprecated in Mac OS X v10.5
Updates the portion of the drag highlight that was drawn over by your application.

## Drag Manager Utilities

`WaitMouseMoved` (page 23)
Returns `true` if a mouse movement is the beginning of a drag.

`ZoomRects` (page 73) Deprecated in Mac OS X v10.5
Animates a rectangle into a second rectangle.

`ZoomRegion` (page 74) Deprecated in Mac OS X v10.5
Animates a region's outline from one screen location to another.

## Creating, Calling, and Deleting Universal Procedure Pointers

`NewDragInputUPP` (page 18)
Creates a new universal procedure pointer (UPP) to a drag input callback.

`DisposeDragInputUPP` (page 12)
Disposes of the universal procedure pointer (UPP) to a drag input callback.

`InvokeDragInputUPP` (page 17)
Calls your drag input callback.

`DisposeDragDrawingUPP` (page 52) Deprecated in Mac OS X v10.5
Disposes of the universal procedure pointer (UPP) to a drag drawing callback.

`DisposeDragReceiveHandlerUPP` (page 53) Deprecated in Mac OS X v10.5
Disposes of the universal procedure pointer (UPP) to a drag receive handler.

`DisposeDragSendDataUPP` (page 53) Deprecated in Mac OS X v10.5
Disposes of the universal procedure pointer (UPP) to a drag send data callback.

`DisposeDragTrackingHandlerUPP` (page 54) Deprecated in Mac OS X v10.5
Disposes of the universal procedure pointer (UPP) to a drag tracking handler.

`InvokeDragDrawingUPP` (page 63) Deprecated in Mac OS X v10.5
Calls your drag drawing callback.

`InvokeDragReceiveHandlerUPP` (page 64) Deprecated in Mac OS X v10.5
Calls your drag receive handler.

`InvokeDragSendDataUPP` (page 64) Deprecated in Mac OS X v10.5
Calls your drag send data callback.

`InvokeDragTrackingHandlerUPP` (page 65) Deprecated in Mac OS X v10.5
Calls your drag tracking handler.

`NewDragDrawingUPP` (page 65) Deprecated in Mac OS X v10.5
Creates a new universal procedure pointer (UPP) to a drag drawing callback.

`NewDragReceiveHandlerUPP` (page 66) Deprecated in Mac OS X v10.5
Creates a new universal procedure pointer (UPP) to a drag receive handler.

`NewDragSendDataUPP` (page 66) Deprecated in Mac OS X v10.5
Creates a new universal procedure pointer (UPP) to a drag send data callback.

`NewDragTrackingHandlerUPP` (page 66) Deprecated in Mac OS X v10.5
Creates a new universal procedure pointer (UPP) to a drag tracking handler.

# Functions

## ChangeDragBehaviors

Changes the behavior of a drag.

```
OSErr ChangeDragBehaviors (
   DragRef theDrag,
   DragBehaviors inBehaviorsToSet,
   DragBehaviors inBehaviorsToClear
);
```

### Parameters

*theDrag*
A drag reference.

*inBehaviorsToSet*
A value indicating the new behavior of the drag. See "Drag Behaviors" (page 36) for a description of the values you can use in this parameter.

*inBehaviorsToClear*
A value indicating which existing behavior, if any, should be cleared. See "Drag Behaviors" (page 36) for a description of the values you can use in this parameter.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## DisposeDrag

Disposes of a drag reference and its associated data.

```
OSErr DisposeDrag (
   DragRef theDrag
);
```

**Parameters**

*theDrag*

The drag reference of the drag object to dispose of. If the drag reference contains any drag item flavors, the memory associated with the drag item flavors is disposed of as well.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You should call the `DisposeDrag` function after a drag has been performed using the `TrackDrag` function or if a drag reference was created but is no longer needed.

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## DisposeDragInputUPP

Disposes of the universal procedure pointer (UPP) to a drag input callback.

```
void DisposeDragInputUPP (
   DragInputUPP userUPP
);
```

**Parameters**

*userUPP*

The UPP to dispose of.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## GetDragAllowableActions

Returns the actions that the drag receiver may take on the data within a drag.

```
OSStatus GetDragAllowableActions (
   DragRef theDrag,
   DragActions *outActions
);
```

**Parameters**

*theDrag*

The drag reference.

*outActions*

A pointer to a field that specifies, on return, the allowable drag actions. See "Drag Actions" (page 40) for a description of the values that may be returned here.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

The drag actions returned by the `GetDragAllowableActions` function are not actually requirements; they are highly recommended suggestions for operations that the drag receiver may perform. The drag sender sets the recommended actions for a drag using the `SetDragAllowableActions` (page 18) function. The drag actions returned by `GetDragAllowableActions` are always local to the caller's process.

**Availability**

Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## GetDragAttributes

Gets the current set of drag attribute flags.

```
OSErr GetDragAttributes (
   DragRef theDrag,
   DragAttributes *flags
);
```

**Parameters**

*theDrag*

A drag reference.

*flags*

On return, a pointer to the drag attribute flags for the specified drag reference. See "Drag Attributes" (page 35) for a description of the values that may be returned here.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

If the `GetDragAttributes` function is called during a drag, the current set of drag attributes is returned. If the `GetDragAttributes` function is called after a drag, the set of drag attributes that were set at drop time is returned.

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**
Drag.h

## GetDragDropAction

Returns the action performed by the receiver of the drag.

```
OSStatus GetDragDropAction (
   DragRef theDrag,
   DragActions *outAction
);
```

**Parameters**

*theDrag*

The drag reference from which to retrieve the drop action.

*outAction*

A pointer to a field that, on return, specifies the action performed by the drag receiver. More than one action may be performed. See "Drag Actions" (page 40) for a description of the values that may be returned here.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

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

**Declared In**
Drag.h

## GetDragItemBounds

Gets the bounding rectangle of a drag item.

```
OSErr GetDragItemBounds (
   DragRef theDrag,
   DragItemRef theItemRef,
   Rect *itemBounds
);
```

**Parameters**

*theDrag*

A drag reference.

*theItemRef*

The reference number of the drag item whose bounds you wish to obtain.

*itemBounds*

On return, a pointer to the bounding rectangle (relative to the current pinned mouse position) of the specified item in global coordinates.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You can use the `GetDragItemBounds` function in your tracking or receive handlers to determine the current or dropped location of each item in the drag.

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetDragModifiers

Gets the current set of keyboard modifiers.

```
OSErr GetDragModifiers (
   DragRef theDrag,
   SInt16 *modifiers,
   SInt16 *mouseDownModifiers,
   SInt16 *mouseUpModifiers
);
```

**Parameters**

*theDrag*
A drag reference.

*modifiers*
A pointer to a variable that, on return, contains the current keyboard modifiers. You may pass `NULL` if you wish to disregard this value. The value will be 0 if the drag has not been started.

*mouseDownModifiers*
A pointer to a variable that, on return, contains the keyboard modifiers at the `mouseDown` parameter time. You may pass `NULL` if you wish to disregard this value. The value will be 0 if the drag has not been started.

*mouseUpModifiers*
A pointer to a variable that, on return, contains the keyboard modifiers at the `mouseUp` parameter time. You may pass `NULL` if you wish to disregard this value. The value will be 0 if the drag has not been completed.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetDragMouse

Gets the current mouse and pinned mouse locations.

```
OSErr GetDragMouse (
   DragRef theDrag,
   Point *mouse,
   Point *globalPinnedMouse
);
```

**Parameters**

*theDrag*

A drag reference.

*mouse*

A pointer to a variable containing, on return, the current mouse location in global screen coordinates. You may pass `NULL` if you wish to ignore this value. The value will be (0, 0) if the drag is not yet used. After a drag completes, the drop location is returned.

*globalPinnedMouse*

A pointer to a variable containing, on return, the current pinned mouse location in global screen coordinates. You may pass `NULL` if you wish to ignore this value. The value will be (0, 0) if the drag is not yet used. After a drag completes, the drop location is returned.The pinned mouse location is the mouse location that is used to draw the drag region on the screen. The pinned mouse location is different from the mouse location when the cursor is being constrained in either dimension by a tracking handler.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetDragOrigin

Gets the `mouseDown` parameter location that started the given drag.

```
OSErr GetDragOrigin (
   DragRef theDrag,
   Point *globalInitialMouse
);
```

**Parameters**

*theDrag*

A drag reference.

*globalInitialMouse*

A pointer to a variable that contains, on return, the `mouseDown` parameter location that started the drag, in global coordinates. The `mouseDown` location is returned whether or not the drag has completed.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## InvokeDragInputUPP

Calls your drag input callback.

```
OSErr InvokeDragInputUPP (
   Point *mouse,
   SInt16 *modifiers,
   void *dragInputRefCon,
   DragRef theDrag,
   DragInputUPP userUPP
);
```

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You should not need to use the function `InvokeDragInputUPP`, as the system calls your drag input callback for you.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## NewDrag

Creates a new drag reference for your application to use with the Drag Manager.

```
OSErr NewDrag (
   DragRef *theDrag
);
```

**Parameters**

*theDrag*

On return, a pointer to the newly created drag reference. This drag reference is required when adding drag item flavors and calling the `TrackDrag` function. Your installed drag handler functions receive this drag reference so they can call other Drag Manager functions.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## NewDragInputUPP

Creates a new universal procedure pointer (UPP) to a drag input callback.

```
DragInputUPP NewDragInputUPP (
   DragInputProcPtr userRoutine
);
```

### Parameters

*userRoutine*

A pointer to your drag input callback.

### Return Value

On return, a UPP to the drag input callback. See the description of the `DragInputUPP` data type.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`Drag.h`

## SetDragAllowableActions

Sets the actions that the drag receiver may take on the data within a drag.

```
OSStatus SetDragAllowableActions (
   DragRef theDrag,
   DragActions inActions,
   Boolean isLocal
);
```

### Parameters

*theDrag*

The drag reference.

*inActions*

A field specifying the allowable actions for the drag. See "Drag Actions" (page 40) for a description of the values you may use here.

*isLocal*

A Boolean value allowing the drag sender to specify whether the actions passed in the `inActions` parameter are allowable for a local receiver or for a remote receiver. Pass `true` in this parameter if the drag actions are for local receivers.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Discussion

The actions set by the drag sender using the `SetDragAllowableActions` function are not requirements; they are highly recommended suggestions for operations the drag receiver may perform. The caller may select whether these drag actions apply to a remote or local process with the `inLocal` parameter.

### Availability

Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## SetDragDropAction

Sets the action performed by the receiver of the drag.

```
OSStatus SetDragDropAction (
   DragRef theDrag,
   DragActions inAction
);
```

**Parameters**

*theDrag*

The drag reference for which to set the drop action.

*inAction*

The drop action performed. More than one action may be performed. See "Drag Actions" (page 40) for a description of the values you may use here.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## SetDragImageWithCGImage

Associates a Core Graphics image with a drag reference.

```
OSStatus SetDragImageWithCGImage (
   DragRef inDrag,
   CGImageRef inCGImage,
   const HIPoint *inImageOffsetPt,
   DragImageFlags inImageFlags
);
```

**Parameters**

*inDrag*

The drag reference for which to display the image.

*inCGImage*

A reference to the image to display during the drag. The Drag Manager retains this image for the duration of the drag, so you may release the image immediately after calling `SetDragImageWithCGImage`.

*inImageOffsetPt*

A pointer to the offset from the mouse location to the upper left corner of the image, normally expressed in negative values. For example, an offset of (-30, -30) centers a 60 by 60 pixel image on the mouse. Note that this differs from the usage of the offset passed to the `SetDragImage` function.

*inImageFlags*

Flags controlling the appearance of the drag image. See "Drag Image Flags" (page 44) for a description of the values you can use in this parameter.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
This function is called by the sender of a drag to set the image displayed to provide user feedback during the drag. You can call the `SetDragImageWithCGImage` function at any point during the drag to update the image.

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

**Declared In**
`Drag.h`

## SetDragInputProc

Sets the drag input function for the Drag Manager to use with a particular drag.

```
OSErr SetDragInputProc (
   DragRef theDrag,
   DragInputUPP inputProc,
   void *dragInputRefCon
);
```

**Parameters**

*theDrag*

The drag reference for which the drag input function will be set.

*inputProc*

The drag input function to be called by the Drag Manager whenever the Drag Manager requires the location of the mouse, the state of the mouse button, and the status of the modifier keys on the keyboard. The Drag Manager typically calls this function once per cycle through the Drag Manager's main drag tracking loop.Your drag input function may either modify the current state of the mouse and keyboard to slightly alter dragging behavior or entirely replace the input data to drive the drag completely by itself. Details for how to write a drag input function are covered in the "Drag Manager Callbacks" (page 24) section.

*dragInputRefCon*

A pointer to a reference constant that will be forwarded to your drag input function when it is called by the Drag Manager. Use this constant to pass any data you wish to forward to your drag input function.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## SetDragItemBounds

Sets the bounding rectangle of a drag item.

```
OSErr SetDragItemBounds (
   DragRef theDrag,
   DragItemRef theItemRef,
   const Rect *itemBounds
);
```

### Parameters

*theDrag*

A drag reference.

*theItemRef*

The reference number of the drag item whose bounds you wish to set.

*itemBounds*

A pointer to the bounding rectangle to set for the specified drag item. This rectangle is specified in global coordinates relative to the mouse down position.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Discussion

Your application would normally call the `SetDragItemBounds` function on each drag item before starting a drag with the `TrackDrag` function.

If you do not set the bounds of an item, the rectangle returned by the `GetDragItemBounds` function is an empty rectangle centered under the pinned mouse location.

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

`Drag.h`

## SetDragMouse

Sets the current pinned mouse location.

```
OSErr SetDragMouse (
   DragRef theDrag,
   Point globalPinnedMouse
);
```

### Parameters

*theDrag*

A drag reference.

*globalPinnedMouse*

The coordinates to which to set the pinned mouse location, in global screen coordinates. The pinned mouse location is used to draw the drag region on the screen.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
To constrain the mouse within one of your application's windows, call the `SetDragMouse` function from within your tracking handler when you receive the `kDragTrackingInWindow` messages. The Drag Manager updates the position of the drag region on the screen after each time your tracking handlers are called.

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## TrackDrag

Drags an item or collection of items from your application.

```
OSErr TrackDrag (
   DragRef theDrag,
   const EventRecord *theEvent,
   RgnHandle theRegion
);
```

**Parameters**

*theDrag*
A drag reference for performing the drag operation.

*theEvent*
A pointer to the `mouseDown` event record that your application received that resulted in starting a drag.

*theRegion*
A region that represents the item or items being dragged. Note that under normal circumstances, the drag region should only include the pixels that represent the outline of the items being dragged. The Drag Manager draws the region on the screen by calling the `PaintRgn` function (not the `FrameRgn` function).

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47). Under some circumstances, `TrackDrag` may fail with a `procNotFound` error. See "Special Considerations" below for a description of the events that may cause this problem.

**Discussion**
The Drag Manager follows the cursor on the screen with the "dotted outline" drag feedback and sends tracking messages to applications that have registered drag tracking handlers. The drag item flavor information cached for the drag is available to each application that becomes active during a drag.

When the user releases the mouse button, the Drag Manager calls any drag receive handlers that have been registered on the destination window. An application's drag receive handler(s) are responsible for accepting the drag and transferring the dragged data into their application.

The `TrackDrag` function returns `noErr` in situations where the user selected a destination for the drag and the destination received data from the Drag Manager. If the user drops over a non-aware application or the receiver does not accept any data from the Drag Manager, the Drag Manager automatically provides a "zoom back" animation and returns the `userCanceledErr` flag.

**Special Considerations**

During the call to the `TrackDrag` function, your application's context is temporarily switched out when the Drag Manager calls a different application's tracking and receive handlers. Do not depend on your application's context to be active for the entire duration of a drag.

The following actions may cause `TrackDrag` to fail with a `procNotFound` error:

- Using a high-level debugger with the Drag Manager. Although there is no workaround for this problem, your code should work fine when run without the debugger.
- Passing `TrackDrag` an event record in which the `where` field is expressed in local coordinates. In such cases, the `where` field often points outside of the window in which the drag originated. This problem can cause a crash as well as a `procNotFound` error.
- Using the Drag Manager with Text Services Manager windows when the `gestaltDragMgrFloatingWind` bit isn't defined.

For more information, see the Q&A at:

http://developer.apple.com/dev/techsupport/develop/issue29/macqa.html.

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## WaitMouseMoved

Returns `true` if a mouse movement is the beginning of a drag.

```
Boolean WaitMouseMoved (
   Point initialGlobalMouse
);
```

**Parameters**

*`initialMouse`*
The point where a `mouseDown` event occurred in global screen coordinates.

**Return Value**
`True` if the mouse moves away from the `initialMouse` location before the mouse button is released, otherwise `false`.

**Discussion**
You can use this function to determine whether you should begin to drag the object when your application receives a `mouseDown` event on a draggable object.

**Availability**
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

# Callbacks by Task

## Tracking and Receiving Drags

`DragTrackingHandlerProcPtr` (page 30)
Defines a pointer to a drag tracking handler.

`DragReceiveHandlerProcPtr` (page 27)
Defines a pointer to a drag receive handler.

## Overriding Drag Manager Behavior

`DragSendDataProcPtr` (page 28)
Defines a pointer to a drag send data function, called by the Drag Manager to supply flavor data to the drag receiver.

`DragInputProcPtr` (page 26)
Defines a pointer to a drag input function that modifies keyboard and mouse input to the Drag Manager.

`DragDrawingProcPtr` (page 24)
Defines a pointer to a drag drawing function that draws the drag region.

# Callbacks

### DragDrawingProcPtr

Defines a pointer to a drag drawing function that draws the drag region.

Not recommended

```
typedef OSErr (*DragDrawingProcPtr) (
                DragRegionMessage message,
                RgnHandle showRegion,
                Point showOrigin,
                RgnHandle hideRegion,
                Point hideOrigin,
                void * dragDrawingRefCon,
                DragRef theDrag);
```

If you name your function `MyDragDrawingFunction`, you would declare it like this:

```
OSErr MyDragDrawingFunction (
                DragRegionMessage message,
                RgnHandle showRegion,
                Point showOrigin,
                RgnHandle hideRegion,
                Point hideOrigin,
                void * dragDrawingRefCon,
```

```
                DragRef theDrag);
```

**Parameters**

*message*

A drag region drawing message from the Drag Manager. Use this message to determine what action your drag drawing callback function should take. These messages are described further in "Drag Drawing Messages" (page 36).

*showRegion*

A region containing the drag region as it should be drawn or is currently visible on the screen, in global screen coordinates. The `showRegion` parameter has slightly different meanings depending on the message passed to your drag drawing callback.

*showOrigin*

The point corresponding to the original `mouseDown` location in the drag region within the given `showRegion`, in global screen coordinates.

*hideRegion*

A region containing the drag region as it should be erased from the screen, in global screen coordinates. The `hideRegion` parameter has slightly different meanings depending on the message passed to your drag drawing callback.

*hideOrigin*

The point corresponding to the original `mouseDown` location in the drag region within the given `hideRegion`, in global screen coordinates.

*dragDrawingRefCon*

A pointer to the reference constant that was provided when the `SetDragDrawingProc` function was called to install this function.

*theDrag*

The drag reference of the drag.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

If your application set a custom drawing function for a drag using the `SetDragDrawingProc` function, the Drag Manager calls this drawing function to perform all drag region drawing operations.

The Drag Manager tracks the drag region as it appears on the screen and as it should follow the mouse. All drag region operations are performed on the region specified to the `TrackDrag` function. Drag region drawing is managed by sending your drag drawing callback function messages to show and hide pieces of the drag region.

The Drag Manager has its own drag region port that is used to do all drag region drawing during a drag. This port is set to the current port before calling your drag drawing function. The drag region port is for your drag drawing function's exclusive use during a drag. You may modify its fields and depend on its contents between calls to your drag drawing callback function.

**Special Considerations**

For Classic applications, your application's context is not available when your drag drawing callback function is called by the Drag Manager. If you need access to your application's global variables, you will need to setup and restore your application's A5 world yourself.

You cannot call the `WaitNextEvent` function or any other Event Manager functions in your drag drawing callback function. This restriction includes calling any functions that may call the Event Manager, such as the `ModalDialog` or `Alert` functions.

#### Carbon Porting Notes

Drag drawing functions are not supported in Mac OS X, although they continue to work in CarbonLib when running Mac OS 8 and Mac OS 9.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

`Drag.h`

## DragInputProcPtr

Defines a pointer to a drag input function that modifies keyboard and mouse input to the Drag Manager.

```
typedef OSErr (*DragInputProcPtr) (

                Point * mouse,
                SInt16 * modifiers,
                void * dragInputRefCon,
                DragRef theDrag);
```

If you name your function `MyDragInputFunction`, you would declare it like this:

```
OSErr MyDragInputFunction (
                Point * mouse,
                SInt16 * modifiers,
                void * dragInputRefCon,
                DragRef theDrag);
```

#### Parameters

*mouse*

On entry, a pointer to the location. On return, your drag input function should provide the desired current mouse location. The mouse location is specified in global coordinates.

*modifiers*

On entry, a pointer to a value indicating the current state of the keyboard modifiers and mouse button. On return, your drag input function should provide a pointer to the desired state of the keyboard modifiers and mouse button. The modifiers are specified using the same format and constants as the Event Manager's `EventRecord.modifiers` field.

*dragInputRefCon*

A pointer to the reference constant that was provided when the `SetDragInputProc` function was called to install this function.

*theDrag*

The drag reference of the drag.

#### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
Each time the Drag Manager samples the mouse and keyboard, it calls your drag input callback (if one has been set by calling the `SetDragInputProc` function) to provide a way for your application to modify or completely change the mouse and keyboard input to the Drag Manager.

When your drag input callback function is called, the mouse and modifiers parameters contain the actual values from the physical input devices. Your drag input callback function may modify these values in any way. For example, your drag input callback function may simply inhibit the control key modifier bit from being set or it may completely replace the mouse coordinates with those generated some other way to drive the drag itself.

Note that the Drag Manager uses the `buttonState` flag in the `modifiers` parameter to determine when the mouse button has been released to finish a drag.

**Special Considerations**

For Classic applications, your application's context is not available when your drag input callback function is called by the Drag Manager. If you need access to your application's global variables, you will need to setup and restore your application's A5 world yourself.

You cannot call the `WaitNextEvent` function or any other Event Manager functions from your drag input callback function. This restriction includes calling any functions that may call the Event Manager, such as the `ModalDialog` or `Alert` functions.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragReceiveHandlerProcPtr

Defines a pointer to a drag receive handler.

```
typedef OSErr (*DragReceiveHandlerProcPtr)
(
                WindowRef theWindow,
                void * handlerRefCon,
                DragRef theDrag);
```

If you name your function `MyDragReceiveHandler`, you would declare it like this:

```
OSErr MyDragReceiveHandler (
                WindowRef theWindow,
                void * handlerRefCon,
                DragRef theDrag);
```

**Parameters**

*`theWindow`*
A reference to the window that the drop occurred in.

*`handlerRefCon`*
A pointer to the reference constant that was provided to the `InstallReceiveHandler` function when this handler was installed.

*theDrag*

The drag reference of the drag.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
When the user releases a drag in a window, the Drag Manager calls any drag receive handler functions that have been installed on that window. You can get the information about the data that is being dragged, to determine if your window will accept the drop, by using the drag information functions provided by the Drag Manager. After your drag receive handler decides that it can accept the drop, use the `GetFlavorData` function to get the data from the sender of the drag.

When the Drag Manager calls your drag receive handler, the port is set to the window that the drop occurred in. If you want to provide an optional Apple Event descriptor of the drop location for the sender, use the `SetDropLocation` function to set the drop location descriptor before calling the sender with the `GetFlavorData` or `GetFlavorDataSize` functions.

If you return any result code other than `noErr` from your drag receive handler, the Drag Manager will "zoomback" the drag region to the source location and return the `userCanceledErr` result code from the `TrackDrag` function. If the drag is dropped into a location that cannot accept the drag (such as the window title bar or window scroll bars) or no acceptable data types were available, your drag receive handler should return the `dragNotAcceptedErr` result code, which will cause the Drag Manager to provide the "zoomback" animation described above.

**Special Considerations**

For Classic applications, the Drag Manager guarantees that your application's A5 world and low-memory environment is properly set up for your application's use. Therefore, you can allocate memory, and use your application's global variables. You can also rely on low-memory globals being valid.

Although it is possible to call `WaitNextEvent` or other functions that run the event loop from within your drag receive handler, it is not recommended as it can cause the drag to timeout and may result in a crash or in corrupt data.

Note that the Process Manager's process switching mechanism is disabled during calls to your handler. If your application is not frontmost when calling these functions, your application will not be able to switch forward. This could result in a situation where a modal dialog appears behind the front process but will not be able to come forward in order to interact with the user.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragSendDataProcPtr

Defines a pointer to a drag send data function, called by the Drag Manager to supply flavor data to the drag receiver.

```
typedef OSErr (*DragSendDataProcPtr)
(
                FlavorType theType,
                void * dragSendRefCon,
                DragItemRef theItemRef,
                DragRef theDrag);
```

If you name your function `MyDragSendDataFunction`, you would declare it like this:

```
OSErr MyDragSendDataFunction (
                FlavorType theType,
                void * dragSendRefCon,
                DragItemRef theItemRef,
                DragRef theDrag);
```

**Parameters**

*theType*

The flavor type being requested by a drop receiver.

*dragSendRefCon*

A pointer to the reference constant that was provided when the `SetDragSendProc` function was called to install this function.

*theItemRef*

The item reference of the item from which the flavor data is being requested.

*theDrag*

The drag reference of the drag.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

The Drag Manager calls your drag send data function when drag item flavor data is requested by a drop receiver if the drag item flavor data is not already cached by the Drag Manager. Use the function `SetDragItemFlavorData` (page 69) to provide the requested data to the Drag Manager.

The Drag Manager caches all drag item flavor data that was specified in the data pointer when the flavor was added using the `AddDragItemFlavor` function. If the data pointer was `NULL` when the flavor was added, the Drag Manager calls the drag send data function to get the data when a receiver requests the data using the `GetFlavorData` or `GetFlavorDataSize` functions.

A second scenario where the drag send data function is called is when a drop receiver requests a flavor that is translated by the Translation Manager and the source data (which would be a different type than actually requested by the receiver) is not already cached by the Drag Manager.

You can use the `GetDropLocation` function to get the Apple event descriptor of the drop location from within your drag send data function. It is optional for the receiver to provide a drop location descriptor. If the receiver does not provide the drop location descriptor, the `typeNull` value will be returned by the `GetDropLocation` function.You do not need to provide a drag send data function if you never pass `NULL` as the data pointer when calling the `AddDragItemFlavor` function.

**Special Considerations**

For Classic applications, the Drag Manager guarantees that your application's A5 world and low-memory environment is properly set up for your application's use. Therefore, you can allocate memory, and use your application's global variables. You can also rely on low-memory globals being valid.

Although it is possible to call `WaitNextEvent` or other functions that run the event loop from within your drag send data callback, it is not recommended as it can cause the drag to timeout and may result in a crash or in corrupt data.

Note that the Process Manager's process switching mechanism is disabled during calls to your handler. If your application is not frontmost when calling these functions, your application will not be able to switch forward. This could result in a situation where a modal dialog appears behind the front process but will not be able to come forward in order to interact with the user.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragTrackingHandlerProcPtr

Defines a pointer to a drag tracking handler.

```
typedef OSErr (*DragTrackingHandlerProcPtr)
(
                DragTrackingMessage message,
                WindowRef theWindow,
                void * handlerRefCon,
                DragRef theDrag);
```

If you name your function `MyDragTrackingHandler`, you would declare it like this:

```
OSErr MyDragTrackingHandler (
        DragTrackingMessage message,
        WindowRef theWindow,
        void * handlerRefCon,
        DragRef theDrag);
```

**Parameters**

*message*
A tracking message from the Drag Manager indicating the action your tracking handler should take. These messages are described further in "Drag Tracking Messages" (page 37).

*theWindow*
A reference to the window that the mouse is currently over.

*handlerRefCon*
A pointer to the reference constant that was provided to the `InstallTrackingHandler` function when this handler was installed.

*theDrag*
The drag reference of the drag.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
When the user drags an item or collection of items through a window, the Drag Manager calls any tracking handlers that have been installed on that window. Your tracking handler can determine the contents of the drag by calling the drag item information functions, such as `CountDragItems` (page 52), `CountDragItemFlavors` (page 51), `GetFlavorType` (page 60) and `GetFlavorFlags` (page 59), and highlighting or modifying the objects in your window accordingly.

When the Drag Manager calls your tracking handler, the port will always be set to the window that the mouse is over.

**Special Considerations**

For Classic applications, the Drag Manager guarantees that your application's A5 world and low-memory environment is properly set up for your application's use. Therefore, you can allocate memory, and use your application's global variables. You can also rely on low-memory globals being valid.

Although it is possible to call `WaitNextEvent` or other functions that run the event loop from within your drag tracking handler, it is not recommended as it can cause the drag to timeout and may result in a crash or in corrupt data.

Note that the Process Manager's process switching mechanism is disabled during calls to your handler. If your application is not frontmost when calling these functions, your application will not be able to switch forward. This could result in a situation where a modal dialog appears behind the front process but will not be able to come forward in order to interact with the user.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

# Data Types

## DragRef

Defines a reference to a drag object.

```
typedef struct OpaqueDragRef * DragRef;
```

**Discussion**
Before calling any other Drag Manager function, you must first create a new drag reference by calling the `NewDrag` function. The drag reference that is returned by the `NewDrag` function is used in all subsequent calls to the Drag Manager. Use the `DisposeDrag` function to dispose of a drag reference after you are finished using it.

The meaning of the bits in a drag reference is internal to the Drag Manager. You should not attempt to interpret the value of the drag reference.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragItemRef

Defines a reference to a drag item.

```
typedef UInt32 DragItemRef;
```

**Discussion**
The drag item reference is a reference number used to refer to a single item in a drag. Drag item reference numbers are created by the sender application when adding drag item flavor information to a drag. Drag item reference numbers are created by and should only be interpreted by the sender application.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## FlavorType

Defines a flavor type.

```
typedef OSType FlavorType;
```

**Discussion**
The flavor type is a four character type that describes the format of drag item flavor data. The flavor type has the same function as a scrap type; it designates the format of the associated data. Any scrap type or resource type may be used.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## HFSFlavor

Defines a flavor for dragging file system objects.

```
struct HFSFlavor {
    OSType fileType;
    OSType fileCreator;
    UInt16 fdFlags;
    FSSpec fileSpec;
};
typedef struct HFSFlavor HFSFlavor;
```

**Fields**

`fileType`
The file type of the object.

`fileCreator`
The file creator of the object.

`fdFlags`
The Finder flags of the object.

`fileSpec`
The `FSSpec` structure for the object.

**Discussion**
The Drag Manager defines a special flavor for dragging file system objects. The HFS drag item flavor is used when dragging document and folder icons in the Finder. The HFS drag item flavor data structure is defined by the `HFSFlavor` data type.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## PromiseHFSFlavor

Defines a data flavor for promising file system objects.

```
struct PromiseHFSFlavor {
    OSType fileType;
    OSType fileCreator;
    UInt16 fdFlags;
    FlavorType promisedFlavor;
};
typedef struct PromiseHFSFlavor PromiseHFSFlavor;
```

**Fields**
`fileType`
The file type of the object.

`fileCreator`
The file creator of the object.

`fdFlags`
The Finder flags of the object.

`promisedFlavor`
The flavor type of a separate promise flavor to contain the `FSSpec` structure for the new file. Apple recommends that you use the `kDragPromisedFlavor` type in this field.

**Discussion**
The promise HFS flavor type is used when you wish to create a new file when dragging to the Finder. The flavor consists of an array of `PromiseHFSFlavor` structures, with the first entry being the preferred file type you would like to create and subsequent array entries being file types in descending preference. This structure allows you to create the file in your `DragSendDataProcPtr` (page 28) callback and provide the `FSSpec` for the new file at that time.

After providing an `FSSpec`, the Finder will move the new file to the drop location. If you wish to create the file before the drag and provide the `FSSpec` data up front, create the new file in the Temporary Items folder so it does not prematurely appear in an open Finder window.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragDrawingUPP

Defines a universal procedure pointer (UPP) to a drag drawing callback.

```
typedef DragDrawingProcPtr DragDrawingUPP;
```

**Discussion**
For more information, see the description of the `DragDrawingProcPtr` (page 24) callback function.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragInputUPP

Defines a universal procedure pointer (UPP) to a drag input callback.

```
typedef DragInputProcPtr DragInputUPP;
```

**Discussion**
For more information, see the description of the `DragInputProcPtr` (page 26) callback function.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragReceiveHandlerUPP

Defines a universal procedure pointer (UPP) to a drag receive handler.

```
typedef DragReceiveHandlerProcPtr DragReceiveHandlerUPP;
```

**Discussion**
For more information, see the description of the `DragReceiveHandlerProcPtr` (page 27) callback function.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## DragSendDataUPP

Defines a universal procedure pointer (UPP) to a drag send data callback.

```
typedef DragSendDataProcPtr DragSendDataUPP;
```

**Discussion**
For more information, see the description of the `DragSendDataProcPtr` (page 28) callback function.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

### DragTrackingHandlerUPP

Defines a universal procedure pointer (UPP) to a drag tracking handler.

```
typedef DragTrackingHandlerProcPtr DragTrackingHandlerUPP;
```

**Discussion**
For more information, see the description of the `DragTrackingHandlerProcPtr` (page 30) callback function.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`Drag.h`

## Constants

### Drag Attributes

Provide additional information about a drag that is in progress.

```
typedef UInt32 DragAttributes;
enum {
    kDragHasLeftSenderWindow = (1L << 0),
    kDragInsideSenderApplication = (1L << 1),
    kDragInsideSenderWindow = (1L << 2) };
```

**Constants**

`kDragHasLeftSenderWindow`
Set if the drag has left the source window since the beginning of the drag. This flag is useful for providing window highlighting after the user has moved the mouse outside of the source window.
Available in Mac OS X v10.0 and later.
Declared in `Drag.h`.

`kDragInsideSenderApplication`
Set if the drag is currently in any window that belongs to the application that started the drag.
Available in Mac OS X v10.0 and later.
Declared in `Drag.h`.

`kDragInsideSenderWindow`
Set if the drag is currently in the same window that the drag started from.
Available in Mac OS X v10.0 and later.
Declared in `Drag.h`.

**Discussion**
The attribute flags defined by the `DragAttributes` type provide information about the window and application that the drag is currently occurring in. During a drag, the current drag attributes can be obtained by calling the function `GetDragAttributes` (page 13).

## Drag Behaviors

Specify the current zoomback behavior of a drag.

```
typedef UInt32 DragBehaviors;
enum {
    kDragBehaviorNone = 0,
    kDragBehaviorZoomBackAnimation = (1L << 0) };
```

**Constants**
`kDragBehaviorNone`
The Drag Manager performs no animation for a failed drag.
Available in Mac OS X v10.0 and later.
Declared in `Drag.h`.

`kDragBehaviorZoomBackAnimation`
The Drag Manager performs zoomback animation for a failed drag. This behavior is normally enabled.
Available in Mac OS X v10.0 and later.
Declared in `Drag.h`.

**Discussion**
To change the behavior associated with a drag reference, use the `ChangeDragBehaviors` (page 11) function.

## Drag Drawing Messages

Define messages that may be sent to your drag drawing callback.

```
typedef SInt16 DragRegionMessage;
enum {
    kDragRegionBegin = 1,
    kDragRegionDraw = 2,
    kDragRegionHide = 3,
    kDragRegionIdle = 4,
    kDragRegionEnd = 5
};
```

**Constants**
`kDragRegionBegin`
Your drag drawing callback function receives this message when a drag is being started and it is time to initialize your drawing function. You should not draw anything to the screen when you receive this message.The `showRegion` and `showOrigin` parameters to your drag drawing callback function contain the drag region and the `mouseDown` location, respectively, that were specified to the `TrackDrag` function. The `mouseDown` location is the origin of the drag region.The `hideRegion` parameter is `NULL` when your drag drawing callback function receives this message.
Available in Mac OS X v10.0 and later.
Declared in `Drag.h`.

`kDragRegionDraw`

Your drag drawing callback receives this message when you should move your drag region from the area of the screen defined by the `hideRegion` parameter to the area of the screen defined by the `showRegion` parameter.The `showRegion` parameter contains the drag region that was passed to the `TrackDrag` function, offset to the current pinned mouse location. This region represents the area of the screen that must be drawn into.The `hideRegion` parameter contains the drag region as it is currently visible on the screen from the last call with a `dragRegionDraw` message. This region represents the area of the screen that must be restored. Any part of the drag region that was previously obscured by a call with the `dragRegionHide` message is not included in this `hideRegion` parameter.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragRegionHide`

Your drag drawing callback receives this message when you should remove part of the drag region from the screen. You receive this message when the drag has ended or when part of the region must be obscured for drawing operations to occur underneath the drag region.The `showRegion` parameter is `NULL` when your drag drawing callback function receives this message.The `hideRegion` parameter contains the part of the currently visible drag region that must be removed from the screen.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragRegionIdle`

Your drag drawing callback receives this message when the drag region has not moved on the screen and no drawing is necessary. You can use this message if animation of the drag region is necessary.The `showRegion` parameter contains the drag region as it is currently visible on the screen.The `hideRegion` parameter is `NULL` when your drag drawing callback receives this message.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragRegionEnd`

Your drag drawing callback receives this message when the drag has completed and it is time to deallocate any allocations made from within your drag drawing callback. Your drag drawing callback will have already received a `dragRegionHide` message to hide the entire drag region before receiving this message. After you receive this message, your drag drawing callback will not be called again for the duration of the drag.Both the `showRegion` and `hideRegion` parameters are `NULL` when your drag drawing callback function receives this message.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

**Discussion**

See `DragDrawingProcPtr` (page 24) for more information on drag drawing callback functions.

## Drag Tracking Messages

Define messages that may be sent to your drag tracking handler.

```
typedef SInt16 DragTrackingMessage;
enum {
    kDragTrackingEnterHandler = 1,
    kDragTrackingEnterWindow = 2,
    kDragTrackingInWindow = 3,
    kDragTrackingLeaveWindow = 4,
    kDragTrackingLeaveHandler = 5
};
```

**Constants**

`kDragTrackingEnterHandler`

Your tracking handler receives this message when the focus of a drag enters a window that is handled by your tracking handler. If the user moves the drag directly to another window that is handled by the same tracking handler, a second `kDragTrackingEnterHandler` message is not received. Your tracking handler only receives this message when the drag enters the domain of your function after leaving another.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragTrackingEnterWindow`

Your tracking handler receives this message when a drag enters any window that is handled by your tracking handler. This message is sent to your tracking handler for each window that the drag may enter. Your tracking handler will always receive this message within a pair of `kDragTrackingEnterHandler` and `kDragTrackingLeaveHandler` messages.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragTrackingInWindow`

Your tracking handler receives this message as the user is dragging within a window handled by your tracking handler. You can use this message to track the dragging process through your window. Your tracking handler will always receive this message within a pair of `kDragTrackingEnterWindow` and `kDragTrackingLeaveWindow` messages.Your tracking handler would typically draw the majority of your window highlighting and track objects in your window when you receive this message from the Drag Manager.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragTrackingLeaveWindow`

Your tracking handler receives this message when a drag leaves any window that is handled by your tracking handler. You are guaranteed to receive this message after receiving a corresponding `kDragTrackingEnterWindow` message. Your tracking handler will always receive this message within a pair of `kDragTrackingEnterHandler` and `kDragTrackingLeaveHandler` messages.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragTrackingLeaveHandler`

Your tracking handler receives this message when the focus of a drag enters a window that is not handled by your tracking handler. Your tracking handler is guaranteed to receive this message after receiving a corresponding `kDragTrackingEnterHandler` message.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

**Discussion**

See `DragTrackingHandlerProcPtr` (page 30) for more information on drag tracking handlers.

## Flavor Flags

Provide additional information about drag item flavors.

```
typedef UInt32 FlavorFlags;
enum {
    flavorSenderOnly = (1 << 0),
    flavorSenderTranslated = (1 << 1),
    flavorNotSaved = (1 << 2),
    flavorSystemTranslated = (1 << 8),
    flavorDataPromised = (1 << 9) };
```

### Constants

`flavorSenderOnly`

Set by the sender if the flavor should only be available to the sender of a drag. If this flag is set when adding the flavor to a drag, no Drag Manager clients other than the sender can receive this flavor.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`flavorSenderTranslated`

Set by the sender if the flavor data is translated by the sender. This flag is useful to a receiver if the receiver needs to determine if the sender is performing its own translation to generate this data type. Typically, receivers that store dragged data without interpreting each data type do not store translated types. Flavor types marked with this flag are not stored by the Finder in clipping files.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`flavorNotSaved`

Set by the sender if the flavor data should not be stored by the receiver. This flag is useful for marking flavor data that will become stale after the drag has completed. Receivers that store dragged data should not store flavors that are marked with this flag. Flavor types marked with this flag are not stored by the Finder in clipping files.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`flavorSystemTranslated`

Set if the flavor data is provided by the Translation Manager. If this flavor is requested, the Drag Manager will obtain any required data types from the sender and then it will use the Translation Manager to provide the data that the receiver requested. Typically, receivers that store dragged data without interpreting each data type do not store translated types. Flavor types marked with this flag are not stored by the Finder in clipping files.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`flavorDataPromised`

Set by the sender if the flavor data is promised at a later time.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

### Discussion

These constants are used when calling the `AddDragItemFlavor` (page 50) function and can be obtained by calling the `GetFlavorFlags` (page 59) function.

## flavorTypeDirectory

Represents a special flavor type for AOCE directory specifications.

```
enum {
    flavorTypeDirectory = 'diry'
};
```

**Constants**

`flavorTypeDirectory`

The flavor type for a AOCE directory specification. Refer to the AOCE documentation for a definition of the `DSSpec` data structure.

Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared in `Drag.h`.

## Drag Actions

Specify the actions that should be or have been performed on the data in a drag.

```
enum {
    kDragActionNothing = 0,
    kDragActionCopy = 1L,
    kDragActionAlias = (1L << 1),
    kDragActionGeneric = (1L << 2),
    kDragActionPrivate = (1L << 3),
    kDragActionMove = (1L << 4),
    kDragActionDelete = (1L << 5),
    kDragActionAll = 0xFFFFFFFF };
typedef UInt32 DragActions;
```

**Constants**

`kDragActionNothing`

Nothing should be or has been done with the data in the drag. When set as an allowable action for remote drags, the drag is not sent to applications other than the drag sender.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionCopy`

The data contained in the drag can be or has been copied.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionAlias`

The data contained in the drag can be or has been shared.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionGeneric`

When set by the drag sender, suggests that the drag receiver can determine the drag action. When returned by the drag receiver, indicates that the receiver did not define a drag action.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionPrivate`
Suggests that the drag action should be negotiated privately between the drag source and destination.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionMove`
The data contained in the drag can be or has been moved.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionDelete`
The data contained in the drag can be or has been deleted.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

`kDragActionAll`
Indicates that all of the above drag actions are allowed.

Available in Mac OS X v10.1 and later.

Declared in `Drag.h`.

**Discussion**
The drag sender can use these constants to indicate what actions are allowable on the data contained within a drag. The drag receiver can use these constants to indicate what, if any, action was performed on the drag.

Some of the drag actions defined here enforce a mode of operation, while others are suggestions. The `DragActions` constants are used in conjunction with the `GetDragAllowableActions` (page 12), `SetDragAllowableActions` (page 18), `GetDragDropAction` (page 14), and `SetDragDropAction` (page 19) functions. Using drag actions increases compatibility with the Cocoa drag operation model.

## HFS Flavor Types

Identify flavor types for file system objects.

```
enum {
    kDragFlavorTypeHFS = 'hfs ',
    kDragFlavorTypePromiseHFS = 'phfs',
    flavorTypeHFS = kDragFlavorTypeHFS,
    flavorTypePromiseHFS = kDragFlavorTypePromiseHFS
};
```

**Constants**
`kDragFlavorTypeHFS`
The flavor type for an HFS file system object. The Finder uses HFS flavors when dragging existing file system objects. The HFS flavor data is defined by the data type `HFSFlavor` (page 32).

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragFlavorTypePromiseHFS`
The flavor type for promising an HFS file system object to the receiver of the drag. This flavor type can be used when a file could be created if the destination of the drag can accept file system objects. The data type `PromiseHFSFlavor` (page 33) is used to access the information in this flavor type.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`flavorTypeHFS`
Use `kDragFlavorTypeHFS` instead.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`flavorTypePromiseHFS`
Use `kDragFlavorTypePromiseHFS` instead.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

## Promised Flavor Types

Identify flavor types for the `PromiseHFSFlavor` structure.

```
enum {
    kDragPromisedFlavorFindFile = 'rWm1',
    kDragPromisedFlavor = 'fssP'
};
```

**Constants**

`kDragPromisedFlavorFindFile`
The value of the `promisedFlavor` field of the `PromiseHFSFlavor` structure for Find File.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragPromisedFlavor`
The value of the `promisedFlavor` field of the `PromiseHFSFlavor` structure for all other file system objects.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

## Type and Creator Constants for Volumes and Directories

Define a creator code and file types for flavor data referring to a volume or directory.

```
enum {
    kDragPseudoCreatorVolumeOrDirectory = 'MACS',
    kDragPseudoFileTypeVolume = 'disk',
    kDragPseudoFileTypeDirectory = 'fold'
};
```

**Constants**

`kDragPseudoCreatorVolumeOrDirectory`
The "creator type" for volumes and directories. If the data in a drag containing `kDragFlavorTypeHFS` data refers to a folder or volume, the `fileCreator` field of the `HFSFlavor` structure should be set to this value.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragPseudoFileTypeVolume`
The value of the `fileType` field of the `HFSFlavor` structure for a volume.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragPseudoFileTypeDirectory`
The value of the `fileType` field of the `HFSFlavor` structure for a directory.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

## Standard Drop Locations

Define common drop locations.

```
enum {
    kDragStandardDropLocationTrash = 'trsh',
    kDragStandardDropLocationUnknown = 'unkn'
};
typedef OSType StandardDropLocation;
```

**Constants**

`kDragStandardDropLocationTrash`
Set when a drag is dropped on the trash icon. Setting this standard drop location automatically sets the traditional drop location to an alias to the trash folder.

Available in Mac OS X v10.2 and later.

Declared in `Drag.h`.

`kDragStandardDropLocationUnknown`
The receiver did not specify a drop location. This is the default.

Available in Mac OS X v10.2 and later.

Declared in `Drag.h`.

**Discussion**

These values are used in conjunction with the `GetStandardDropLocation` (page 60) and `SetStandardDropLocation` (page 71) functions.

## Drag Image Flags

Specify the appearance of a translucent drag.

```
typedef UInt32 DragImageFlags;
enum {
    kDragRegionAndImage = (1L << 4)
    kDragStandardTranslucency = 0,
    kDragDarkTranslucency = 1,
    kDragDarkerTranslucency = 2,
    kDragOpaqueTranslucency = 3
};
```

**Constants**

`kDragRegionAndImage`

Add this constant to the transparency levels represented by the following constants to specify that the outline region passed to `TrackDrag` should be drawn on screen, in addition to the translucent drag image.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragStandardTranslucency`

Use the standard translucency level for the drag image. Currently, this is 65%.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragDarkTranslucency`

Use 50% translucency for the drag image.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragDarkerTranslucency`

Use 25% transparency for the drag image.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kDragOpaqueTranslucency`

Use an opaque drag image (0% translucency).

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

**Discussion**

These constants are used in conjunction with the `SetDragImageWithCGImage` (page 19)and `SetDragImage` (page 49) functions to specify the appearance of the drag image.

## Finder Flavor Types

Identify flavor types for the Finder.

```
enum {
    kFlavorTypeClippingName = 'clnm',
    kFlavorTypeClippingFilename = 'clfn',
    kFlavorTypeUnicodeClippingName = 'ucln',
    kFlavorTypeUnicodeClippingFilename = 'uclf',
    kFlavorTypeDragToTrashOnly = 'fdtt',
    kFlavorTypeFinderNoTrackingBehavior = 'fntb'
};
```

**Constants**

`kFlavorTypeClippingName`
The flavor of a name hint for a clipping file. This flavor type is preferred over the `kFlavorTypeClippingFilename` type.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kFlavorTypeClippingFilename`
The flavor of the name of a clipping file.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kFlavorTypeUnicodeClippingName`
The flavor of a hint for the unicode name of a clipping file. This flavor type is preferred over the `kFlavorTypeUnicodeClippingFilename` type.

Available in Mac OS X v10.2 and later.

Declared in `Drag.h`.

`kFlavorTypeUnicodeClippingFilename`
The flavor of the unicode name of a clipping file.

Available in Mac OS X v10.2 and later.

Declared in `Drag.h`.

`kFlavorTypeDragToTrashOnly`
Specify this flavor to allow dragging private data to the trash.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kFlavorTypeFinderNoTrackingBehavior`
A flavor type indicating that the Finder should ignore the drag.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

## Zoom Acceleration Constants

Specify acceleration constants for the `ZoomRects` and `ZoomRegion` functions.

```
typedef SInt16 ZoomAcceleration;
enum {
    kZoomNoAcceleration = 0,
    kZoomAccelerate = 1,
    kZoomDecelerate = 2
};
```

**Constants**

`kZoomNoAcceleration`

Use linear interpolation for each frame of animation between the source and destination.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kZoomAccelerate`

Increment the step size for each frame of animation between the source and destination. This option produces the visual appearance of the animation speeding up as it approaches the destination.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

`kZoomDecelerate`

Decrement the step size for each frame of animation between the source and destination. This option produces the visual appearance of the animation slowing down as it approaches the destination.

Available in Mac OS X v10.0 and later.

Declared in `Drag.h`.

## zoomNoAcceleration

Obsolete. Use "Zoom Acceleration Constants" instead.

```
enum {
    zoomNoAcceleration = kZoomNoAcceleration,
    zoomAccelerate = kZoomAccelerate,
    zoomDecelerate = kZoomDecelerate
};
```

## kDragStandardImage

Obsolete. Use "Drag Image Flags" instead.

```
enum {
    kDragStandardImage = kDragStandardTranslucency,
    kDragDarkImage = kDragDarkTranslucency,
    kDragDarkerImage = kDragDarkerTranslucency,
    kDragOpaqueImage = kDragOpaqueTranslucency
};
```

## dragTrackingEnterHandler

Obsolete. Use "Drag Tracking Messages" instead.

```
enum {
    dragTrackingEnterHandler = kDragTrackingEnterHandler,
    dragTrackingEnterWindow = kDragTrackingEnterWindow,
    dragTrackingInWindow = kDragTrackingInWindow,
    dragTrackingLeaveWindow = kDragTrackingLeaveWindow,
    dragTrackingLeaveHandler = kDragTrackingLeaveHandler
};
```

### dragRegionBegin

Obsolete. Use "Drag Drawing Messages" instead.

```
enum {
    dragRegionBegin = kDragRegionBegin,
    dragRegionDraw = kDragRegionDraw,
    dragRegionHide = kDragRegionHide,
    dragRegionIdle = kDragRegionIdle,
    dragRegionEnd = kDragRegionEnd
};
```

### dragHasLeftSenderWindow

Obsolete. Use "Drag Attributes" instead.

```
enum {
    dragHasLeftSenderWindow = kDragHasLeftSenderWindow,
    dragInsideSenderApplication = kDragInsideSenderApplication,
    dragInsideSenderWindow = kDragInsideSenderWindow
};
```

## Result Codes

The table below lists the most common result codes returned by the Drag Manager.

| Result Code | Value | Description |
| --- | --- | --- |
| `badDragRefErr` | -1850 | Unknown drag reference<br>Available in Mac OS X v10.0 and later. |
| `badDragItemErr` | -1851 | Unknown drag item reference<br>Available in Mac OS X v10.0 and later. |
| `badDragFlavorErr` | -1852 | Unknown flavor type<br>Available in Mac OS X v10.0 and later. |
| `duplicateFlavorErr` | -1853 | Flavor type already exists<br>Available in Mac OS X v10.0 and later. |

| Result Code | Value | Description |
|---|---|---|
| `cantGetFlavorErr` | -1854 | Error while trying to get flavor data<br>Available in Mac OS X v10.0 and later. |
| `duplicateHandlerErr` | -1855 | Handler already exists<br>Available in Mac OS X v10.0 and later. |
| `handlerNotFoundErr` | -1856 | Handler not found<br>Available in Mac OS X v10.0 and later. |
| `dragNotAcceptedErr` | -1857 | Drag was not accepted by receiver<br>Available in Mac OS X v10.0 and later. |
| `unsupportedForPlatformErr` | -1858 | Call is for PowerPC only<br>Available in Mac OS X v10.0 and later. |
| `noSuitableDisplaysErr` | -1859 | No displays support translucency<br>Available in Mac OS X v10.0 and later. |
| `badImageRgnErr` | -1860 | Bad translucent image region<br>Available in Mac OS X v10.0 and later. |
| `badImageErr` | -1861 | Bad translucent image PixMap<br>Available in Mac OS X v10.0 and later. |
| `nonDragOriginatorErr` | -1862 | Illegal attempt to access originator only data<br>Available in Mac OS X v10.0 and later. |

# Gestalt Constants

You can check for version and feature availability information by using the Drag Manager selectors defined in the Gestalt Manager. For more information see *Inside Mac OS X: Gestalt Manager Reference*

# Deprecated Drag Manager Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

## Deprecated in Mac OS X v10.4

### SetDragImage

Associates an image with a drag reference. (Deprecated in Mac OS X v10.4. Use `SetDragImageWithCGImage` (page 19) instead.)

```
OSErr SetDragImage (
   DragRef inDrag,
   PixMapHandle inImagePixMap,
   RgnHandle inImageRgn,
   Point inImageOffsetPt,
   DragImageFlags inImageFlags
);
```

#### Parameters

*theDrag*

The drag reference.

*imagePixMap*

A handle to a `PixMap` describing the image. The Drag Manager temporarily locks the `PixMapHandle` during the drag. The Drag Manager does not copy the information in this parameter; you must ensure that the data to which this parameter refers continues to exist until `TrackDrag` completes.

*imageRgn*

A mask describing the portion of the `PixMap` contained in the `imagePixMap` parameter which contains the drag image. Pass `NULL` for `imageRgn` if the entire `PixMap`, including white space, should be dragged.

The Drag Manager does not copy the `imageRgn` parameter data. Until `TrackDrag` completes or `SetDragImage` is called again to update the image, you must ensure that the data to which this parameter refers continues to exist.

Don't confuse the region passed to the function `TrackDrag` and that passed to the `SetDragImage` function. The former is what's drawn to the screen during dragging, while the latter is used only for drawing the correct portion of the drag image.

*imageOffsetPt*

The offset required to move the `PixMap` specified in the `imagePixMap` parameter to the global coordinates where the image initially appears. If this parameter is (0,0), the `PixMap` should already be in global coordinates.

*theImageFlags*

Flags controlling the appearance of the drag image. See "Drag Image Flags" (page 44) for a description of the values you can use in this parameter.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
The sending application should call `SetDragImage` prior to calling `TrackDrag`. Prior to calling `SetDragImage`, the application should draw a solid, opaque image into the `PixMap` specified in the `imagePixMap` parameter. The Drag Manager will provide translucency effects. Your application can obtain a `PixMap` by calling the QuickDraw function `GetGWorldPixMap` and supplying a `GWorld` into which your application has drawn the image.

To allow the Drag Manager to analyze the `PixMap`'s colors in order to determine if it can be rendered on the available screens, Apple recommends using an 8-bit `GWorld` for the `PixMap`.

**Special Considerations**

`SetDragImage` installs a custom drawing procedure to do the translucent drawing. Applications calling `SetDragImage` should not also call `SetDragDrawingProc` for the same drag.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

# Deprecated in Mac OS X v10.5

## AddDragItemFlavor

Adds a flavor to a drag item, creating a new item if necessary. (Deprecated in Mac OS X v10.5.)

```
OSErr AddDragItemFlavor (
   DragRef theDrag,
   DragItemRef theItemRef,
   FlavorType theType,
   const void *dataPtr,
   Size dataSize,
   FlavorFlags theFlags
);
```

**Parameters**
*theDrag*
A drag reference.

*theItemRef*

The drag item to add the flavor to. You create a new drag item by providing a unique item reference number here. You add a flavor to an existing item by using the same item reference number as in a previous call. You may use any item reference number when adding a flavor to an item. Item reference numbers do not need to be specified in order, nor must they be sequential. In many cases it is easiest to use index numbers as item reference numbers (1, 2, 3...). Item reference numbers are only used as unique "key" numbers for each item. Depending on your application, it might be easier to use your own internal memory addresses as item reference numbers (as long as each item being dragged has a unique item reference number).

*theType*

The data type of the flavor to add. This may be any four-character scrap type. You may use your application's signature for a unique type for internal use. You must add all of the drag item flavors to a drag item before calling the `TrackDrag` function. Once the `TrackDrag` function is called, receiving applications may not operate properly if new drag items or drag item flavors are added.

*dataPtr*

A pointer to the flavor data to add. Pass `NULL` to defer the creation of a particular data type until a receiver has specifically requested it. If you pass `NULL`, a promise is added to the drag; when the flavor is requested, the Drag Manager calls the drag's send data function to get the data from your application.

Note that this method of setting promises differs from the method of setting Scrap Manager promises. See the Scrap Manager function `PutScrapFlavor` for more information.

*dataSize*

The size, in bytes, of the flavor data to add. If you pass `NULL` in the `dataPtr` parameter, the value in this parameter is ignored.

*theFlags*

The set of attributes to set for this flavor.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## CountDragItemFlavors

Gets the number of flavors that are contained within a drag item. (Deprecated in Mac OS X v10.5.)

```
OSErr CountDragItemFlavors (
   DragRef theDrag,
   DragItemRef theItemRef,
   UInt16 *numFlavors
);
```

**Parameters**

*theDrag*

The drag reference.

*theItemRef*
An item reference number.

*numFlavors*
On return, a pointer to the number of flavors in the specified drag item. When the `CountDragItemFlavors` function is called by an application other than the sender, the flavors that are marked with the `flavorSenderOnly` flag are not included in the count.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## CountDragItems

Gets the number of drag items that are contained in a drag reference. (Deprecated in Mac OS X v10.5.)

```
OSErr CountDragItems (
   DragRef theDrag,
   UInt16 *numItems
);
```

**Parameters**

*theDrag*
The drag reference.

*numItems*
On return, a pointer to the number of drag items in the specified drag reference.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## DisposeDragDrawingUPP

Disposes of the universal procedure pointer (UPP) to a drag drawing callback. (Deprecated in Mac OS X v10.5.)

```
void DisposeDragDrawingUPP (
   DragDrawingUPP userUPP
);
```

**Parameters**

*userUPP*

The UPP to dispose of.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

**Declared In**

`Drag.h`

## DisposeDragReceiveHandlerUPP

Disposes of the universal procedure pointer (UPP) to a drag receive handler. (Deprecated in Mac OS X v10.5.)

```
void DisposeDragReceiveHandlerUPP (
   DragReceiveHandlerUPP userUPP
);
```

**Parameters**

*userUPP*

The UPP to dispose of.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

**Declared In**

`Drag.h`

## DisposeDragSendDataUPP

Disposes of the universal procedure pointer (UPP) to a drag send data callback. (Deprecated in Mac OS X v10.5.)

```
void DisposeDragSendDataUPP (
   DragSendDataUPP userUPP
);
```

**Parameters**

*userUPP*

The UPP to dispose of.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

**Declared In**

`Drag.h`

## DisposeDragTrackingHandlerUPP

Disposes of the universal procedure pointer (UPP) to a drag tracking handler. (Deprecated in Mac OS X v10.5.)

```
void DisposeDragTrackingHandlerUPP (
   DragTrackingHandlerUPP userUPP
);
```

### Parameters

*userUPP*

The UPP to dispose of.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

### Declared In

`Drag.h`

## DragPostScroll

Restores the drag highlight after scrolling part of your window. (Deprecated in Mac OS X v10.5.)

```
OSErr DragPostScroll (
   DragRef theDrag
);
```

### Parameters

*theDrag*

The drag reference.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Discussion

The `DragPostScroll` function restores the drag highlight after scrolling part of your window. This function must be called following each call to the `DragPreScroll` function and any subsequent scrolling.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

### Declared In

`Drag.h`

## DragPreScroll

Prepares your window or pane for scrolling. (Deprecated in Mac OS X v10.5.)

```
OSErr DragPreScroll (
   DragRef theDrag,
   SInt16 dH,
   SInt16 dV
);
```

**Parameters**

*theDrag*

The drag reference.

*dH*

The horizontal distance you intend to scroll.

*dV*

The vertical distance you intend to scroll.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

Removes any drag highlighting that would be scrolled away from the `hiliteFrame` specified to the `ShowDragHilite` function when scrolling part of your window while drag highlighting is showing. Use this function if you plan to scroll part of your window using the `ScrollRect` or `CopyBits` functions.

Scrolling part of your window may inadvertently move part of the drag highlighting with it. The `DragPreScroll` function is optimized to remove from the screen only the parts of the highlighting that will be scrolled away from the `hiliteFrame` region. After calling the `DragPreScroll` function with the `dH` and `dV` that you are going to scroll, you can then scroll your window followed by a call to the `DragPostScroll` function which redraws any necessary highlighting after the scroll.

If you use an offscreen port to draw your window into while scrolling, it is recommended that you simply use the `HideDragHilite` and `ShowDragHilite` functions to preserve drag highlighting in your offscreen port. The `DragScroll` functions are optimized for onscreen scrolling.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## GetDragHiliteColor

Returns the drag highlight color for a window. (Deprecated in Mac OS X v10.5.)

```
OSErr GetDragHiliteColor (
   WindowRef window,
   RGBColor *color
);
```

**Parameters**

*window*

The window for which to return the drag highlight color.

*color*
On return, a pointer to the highlight color.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
Use the `GetDragHiliteColor` function to determine the color the Drag Manager will use for a particular window. `GetDragHiliteColor` can safely be called when the `gestaltDragMgrHasImageSupport` bit is set in the `Gestalt` response to the selector `gestaltDragMgrAttr`. For more information on the `gestaltDragMgrAttr` selector, see *Inside Mac OS X: Gestalt Manager Reference*.

The Drag Manager chooses an appropriate color for highlighting, based on the color used for drag highlighting in the current Appearance Manager theme.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetDragItemReferenceNumber

Gets the reference number of a specific item in a drag reference. (Deprecated in Mac OS X v10.5.)

```
OSErr GetDragItemReferenceNumber (
   DragRef theDrag,
   UInt16 index,
   DragItemRef *theItemRef
);
```

**Parameters**
*theDrag*
The drag reference.

*index*
The index of an item in a drag for which to get the reference.

*theItemRef*
On return, a pointer to the reference number of the item with the specified index.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47). If `index` is 0 or larger than the number of items in the drag, `GetDragItemReferenceNumber` returns the `badDragItemErr` result code.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetDropLocation

Gets the Apple Event descriptor of the drop location. (Deprecated in Mac OS X v10.5.)

```
OSErr GetDropLocation (
   DragRef theDrag,
   AEDesc *dropLocation
);
```

### Parameters

*theDrag*

A drag reference.

*dropLocation*

On return, a pointer to the Apple Event descriptor of the drop location. The drop location is only valid after the receiver has set the drop location by calling the `SetDropLocation` function. If the destination is in the Finder, the drop location will be an alias to the location in the file system that received the drag. If the receiver of the drag has not set a drop location by calling the `SetDropLocation` function, `typeNull` will be returned.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Discussion

The `GetDropLocation` function may be called both during a drag as well as after a drag has completed.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

### Declared In

`Drag.h`

## GetFlavorData

Gets all or part of the data for a specific flavor in a drag item. (Deprecated in Mac OS X v10.5.)

```
OSErr GetFlavorData (
   DragRef theDrag,
   DragItemRef theItemRef,
   FlavorType theType,
   void *dataPtr,
   Size *dataSize,
   UInt32 dataOffset
);
```

### Parameters

*theDrag*

A drag reference.

*theItemRef*

The reference number of the drag item containing the flavor data.

*theType*

The flavor type of the flavor to get the data from.

*dataPtr*

A pointer to a data buffer. On return, the buffer contains the requested flavor data. Your application is responsible for allocating the memory for the flavor data and for setting the *dataSize* parameter to the number of bytes that you have allocated for the data.

*dataSize*

On input, a pointer to the size of the data (in bytes) that you have allocated memory for and wish to receive from the flavor. On return, a pointer to the actual number of bytes copied into the buffer specified by the `dataPtr` parameter.

If you specify a `dataSize` that is smaller than the amount of data in the flavor, the data is copied into your buffer and the `dataSize` parameter is unchanged. If you specify a `dataSize` that is larger than the amount of data in the flavor, only the amount of data in the flavor is copied into your buffer and the `dataSize` parameter contains, on return, the actual number of bytes copied. You have reached the end of the flavor's data when the `dataSize` parameter points to a number of bytes lower than you provided.

If you wish to receive the flavor data in smaller pieces than the entire size of the data, you can set the `dataSize` parameter to be as large as your buffer and call the `GetFlavorData` function multiple times while incrementing the `dataOffset` parameter by the size of your buffer. If the `dataOffset` parameter is larger than the amount of data contained within the flavor, 0 (zero) will be returned in the number pointed to by the `dataSize` parameter indicating that no data was copied into your buffer.

*dataOffset*

A pointer to the offset (in bytes) within the flavor structure at which to begin copying data.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You can first determine the size of a flavor's data by calling the `GetFlavorDataSize` function.

Note that calling the `GetFlavorData` function on a flavor that requires translation will force that translation to occur in order to return the data.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetFlavorDataSize

Gets the size of the data for a specific flavor in a drag item. (Deprecated in Mac OS X v10.5.)

```
OSErr GetFlavorDataSize (
   DragRef theDrag,
   DragItemRef theItemRef,
   FlavorType theType,
   Size *dataSize
);
```

**Parameters**

*theDrag*

A drag reference.

*theItemRef*

The reference number of the drag item containing the flavor.

*theType*

The flavor type for which to get the size of the data.

*dataSize*

On return, a pointer to the size of the data for the specified drag item flavor.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

Note that calling the `GetFlavorDataSize` function on a flavor that requires translation will force that translation to be performed in order to determine the data size. Since translation may require a significant amount of time and memory during processing, call the `GetFlavorDataSize` function only when necessary.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## GetFlavorFlags

Gets the flags for a specific flavor in a drag item. (Deprecated in Mac OS X v10.5.)

```
OSErr GetFlavorFlags (
   DragRef theDrag,
   DragItemRef theItemRef,
   FlavorType theType,
   FlavorFlags *theFlags
);
```

**Parameters**

*theDrag*

A drag reference.

*theItemRef*

The reference number of the drag item containing the flavor.

*theType*

The flavor type for which to get the attributes.

*theFlags*
On return, a pointer to the attributes of the specified flavor. If a flavor is marked with the `flavorSenderOnly` flag, it is not returned to any application other than the sender.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetFlavorType

Gets the type of a specific flavor in a drag item. (Deprecated in Mac OS X v10.5.)

```
OSErr GetFlavorType (
   DragRef theDrag,
   DragItemRef theItemRef,
   UInt16 index,
   FlavorType *theType
);
```

**Parameters**
*theDrag*
A drag reference.
*theItemRef*
The reference number of the drag item containing the flavor.
*index*
The index of the desired flavor.
*theType*
On return, a pointer to the type of the specified flavor. If a flavor is marked with the `flavorSenderOnly` flag, it is not returned to any application other than the sender.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47). If `index` is 0 or larger than the number of flavors in the item, `GetFlavorType` returns the `badDragFlavorErr` result code.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## GetStandardDropLocation

Gets the standard drop location set by the receiver of a drag. (Deprecated in Mac OS X v10.5.)

```
OSStatus GetStandardDropLocation (
   DragRef theDrag,
   StandardDropLocation *outDropLocation
);
```

**Parameters**

*theDrag*

The drag reference.

*outDropLocation*

A pointer to a value that, on return, represents the location where the drag was dropped. You can use the `GetStandardDropLcoation` function to easily determine whether a drag landed in the trash; if the drop location is the trash, the value of this parameter is `kDragStandardDropLocationTrash`. Otherwise, the value returned here is `kDragStandardDropLocationUnknown`. See "Standard Drop Locations" (page 43) for more information on these values.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Availability**

Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## HideDragHilite

Removes highlighting created with the `ShowDragHilite` function. (Deprecated in Mac OS X v10.5.)

```
OSErr HideDragHilite (
   DragRef theDrag
);
```

**Parameters**

*theDrag*

The drag reference that is currently showing a drag highlight.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

The `HideDragHilite` function assumes that the highlighting should be erased from the current port. Your application should make sure that the correct port is set before calling the `HideDragHilite` function. Also, highlighting erased by the `HideDragHilite` function is clipped to the current port. Make sure that the port's clip region is appropriately sized to remove the highlighting.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## InstallReceiveHandler

Installs a receive handler function for one or all of your application's windows. (Deprecated in Mac OS X v10.5.)

```
OSErr InstallReceiveHandler (
   DragReceiveHandlerUPP receiveHandler,
   WindowRef theWindow,
   void *handlerRefCon
);
```

**Parameters**

*receiveHandler*

A pointer to a receive handler function. Installing a receive handler function allows your application to accept a drag by getting drag item flavor data from the Drag Manager when the user releases the mouse button while dragging over one of your application's windows.

*theWindow*

A reference to the window for which to install the receive handler. When a drop occurs over this window, the Drag Manager calls your receive handler function to allow your application to accept the drag. If you pass `NULL`, the receive handler function is installed in the default handler space for your application. Receive handler functions installed in this way are called when a drop occurs over any window that belongs to your application. You may install more than one receive handler function on a single window.

*handlerRefCon*

A pointer to a reference constant that will be forwarded to your receive handler function when it is called by the Drag Manager. Use this constant to pass any data you wish to forward to your drag receive handler.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
The Drag Manager sequentially calls all of the receive handler functions installed on a window when a drop occurs in that window.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## InstallTrackingHandler

Installs a tracking handler function for one or all of your application's windows. (Deprecated in Mac OS X v10.5.)

```
OSErr InstallTrackingHandler (
   DragTrackingHandlerUPP trackingHandler,
   WindowRef theWindow,
   void *handlerRefCon
);
```

**Parameters**

*trackingHandler*

A pointer to a tracking handler function. Installing a tracking handler function allows your application to track the user's movements through your application's windows during a drag.

*theWindow*

A reference to the window for which to track and handle dragging. When the cursor moves into this window during a drag, the Drag Manager sends tracking messages to the tracking handler function. If you pass `NULL`, the tracking handler function is installed in the default handler space for your application. Tracking handler functions installed in this way are called when the user moves the mouse over any window that belongs to your application. You may install more than one drag tracking handler on a single window.

*handlerRefCon*

A pointer to a reference constant that will be forwarded to your tracking handler function when it is called by the Drag Manager. Use this constant to pass any data you wish to forward to your tracking handler function.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
The Drag Manager sequentially calls all of the tracking handler functions installed for a window when the user moves the cursor over that window during a drag.

**Availability**
Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## InvokeDragDrawingUPP

Calls your drag drawing callback. (Deprecated in Mac OS X v10.5.)

```
OSErr InvokeDragDrawingUPP (
   DragRegionMessage message,
   RgnHandle showRegion,
   Point showOrigin,
   RgnHandle hideRegion,
   Point hideOrigin,
   void *dragDrawingRefCon,
   DragRef theDrag,
   DragDrawingUPP userUPP
);
```

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You should not need to use the function `InvokeDragDrawingUPP`, as the system calls your drag drawing callback for you.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

**Declared In**
`Drag.h`

## InvokeDragReceiveHandlerUPP

Calls your drag receive handler. (Deprecated in Mac OS X v10.5.)

```
OSErr InvokeDragReceiveHandlerUPP (
   WindowRef theWindow,
   void *handlerRefCon,
   DragRef theDrag,
   DragReceiveHandlerUPP userUPP
);
```

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You should not need to use the function `InvokeDragReceiveHandlerUPP`, as the system calls your drag receive handler for you.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

**Declared In**
`Drag.h`

## InvokeDragSendDataUPP

Calls your drag send data callback. (Deprecated in Mac OS X v10.5.)

```
OSErr InvokeDragSendDataUPP (
   FlavorType theType,
   void *dragSendRefCon,
   DragItemRef theItemRef,
   DragRef theDrag,
   DragSendDataUPP userUPP
);
```

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You should not need to use the function `InvokeDragSendDataUPP`, as the system calls your drag send data callback for you.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

**Declared In**
`Drag.h`

## InvokeDragTrackingHandlerUPP

Calls your drag tracking handler. (Deprecated in Mac OS X v10.5.)

```
OSErr InvokeDragTrackingHandlerUPP (
   DragTrackingMessage message,
   WindowRef theWindow,
   void *handlerRefCon,
   DragRef theDrag,
   DragTrackingHandlerUPP userUPP
);
```

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
You should not need to use the function `InvokeDragTrackingHandlerUPP`, as the system calls your drag tracking handler for you.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

**Declared In**
`Drag.h`

## NewDragDrawingUPP

Creates a new universal procedure pointer (UPP) to a drag drawing callback. (Deprecated in Mac OS X v10.5.)

```
DragDrawingUPP NewDragDrawingUPP (
   DragDrawingProcPtr userRoutine
);
```

**Parameters**
*`userRoutine`*
A pointer to your drag drawing callback.

**Return Value**
On return, a UPP to the drag drawing callback. See the description of the `DragDrawingUPP` data type.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.

**Declared In**
`Drag.h`

## NewDragReceiveHandlerUPP

Creates a new universal procedure pointer (UPP) to a drag receive handler. (Deprecated in Mac OS X v10.5.)

```
DragReceiveHandlerUPP NewDragReceiveHandlerUPP (
   DragReceiveHandlerProcPtr userRoutine
);
```

### Parameters

*userRoutine*

A pointer to your drag receive handler.

### Return Value

On return, a UPP to the drag receive handler. See the description of the `DragReceiveHandlerUPP` data type.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

### Declared In

`Drag.h`

## NewDragSendDataUPP

Creates a new universal procedure pointer (UPP) to a drag send data callback. (Deprecated in Mac OS X v10.5.)

```
DragSendDataUPP NewDragSendDataUPP (
   DragSendDataProcPtr userRoutine
);
```

### Parameters

*userRoutine*

A pointer to your drag send data callback.

### Return Value

On return, a UPP to the drag send data callback. See the description of the `DragSendDataUPP` data type.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

### Declared In

`Drag.h`

## NewDragTrackingHandlerUPP

Creates a new universal procedure pointer (UPP) to a drag tracking handler. (Deprecated in Mac OS X v10.5.)

```
DragTrackingHandlerUPP NewDragTrackingHandlerUPP (
   DragTrackingHandlerProcPtr userRoutine
);
```

**Parameters**

*userRoutine*

A pointer to your drag tracking handler.

**Return Value**

On return, a UPP to the drag tracking handler. See the description of the `DragTrackingHandlerUPP` data type.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

**Declared In**

`Drag.h`

## RemoveReceiveHandler

Removes a receive handler function from one or all of your application's windows. (Deprecated in Mac OS X v10.5.)

```
OSErr RemoveReceiveHandler (
   DragReceiveHandlerUPP receiveHandler,
   WindowRef theWindow
);
```

**Parameters**

*receiveHandler*

A pointer to a receive handler function.

*theWindow*

A reference to the window from which to remove the receive handler function. Pass `NULL` to remove the specified receive handler function from the default handler space for your application.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## RemoveTrackingHandler

Removes a tracking handler function from one or all of your application's windows. (Deprecated in Mac OS X v10.5.)

```
OSErr RemoveTrackingHandler (
   DragTrackingHandlerUPP trackingHandler,
   WindowRef theWindow
);
```

**Parameters**

*trackingHandler*

A pointer to the tracking handler function to be removed.

*theWindow*

A reference to the window from which to remove the drag tracking handler function. Pass `NULL` to remove the specified tracking handler function from the default handler space for your application.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## SetDragDrawingProc

Sets the drag drawing function for the Drag Manager to use with a particular drag. (Deprecated in Mac OS X v10.5.)

Not recommended

```
OSErr SetDragDrawingProc (
   DragRef theDrag,
   DragDrawingUPP drawingProc,
   void *dragDrawingRefCon
);
```

**Parameters**

*theDrag*

The drag reference for which the drag drawing function will be set.

*drawingProc*

The drag drawing function to be called by the Drag Manager to draw, move, and hide the "dotted outline" drag feedback on the screen during a drag.Your drag drawing function can implement any type of drag feedback, such as dragging a bitmap of the object being dragged. Details for how to write a drag drawing function are covered in the "Drag Manager Callbacks" (page 24) section.

*dragDrawingRefCon*

A pointer to a reference constant that will be forwarded to your drag drawing function when it is called by the Drag Manager. Use this constant to pass any data you wish to forward to your drag drawing function.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Carbon Porting Notes**

Drag drawing functions are not supported in Mac OS X, although they continue to work in CarbonLib when running Mac OS 8 and Mac OS 9.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## SetDragItemFlavorData

Sets the data or part of the data contained within an existing flavor. (Deprecated in Mac OS X v10.5.)

```
OSErr SetDragItemFlavorData (
   DragRef theDrag,
   DragItemRef theItemRef,
   FlavorType theType,
   const void *dataPtr,
   Size dataSize,
   UInt32 dataOffset
);
```

**Parameters**

*theDrag*
The drag reference whose flavor data will be set.

*theItemRef*
The drag item reference of the item that contains the flavor you wish to set all or part of the data for. The data pointed to by the *dataPtr* parameter with the size specified in the *dataSize* parameter is placed into the flavor structure at the offset specified by the *dataOffset* parameter.

*theType*
The data type of the existing flavor for which all or part of the data will be set.

*dataPtr*
A pointer to the flavor data.

*dataSize*
The size, in bytes, of the flavor data.

*dataOffset*
The offset, in bytes, into the flavor structure at which to place the data specified by the *dataPtr* and the *dataSize* parameters.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
This function is commonly used in situations where a flavor's data is not added to the flavor when the flavor is created using the `AddDragItemFlavor` function. When the sender's drag send data function is called, the `SetDragItemFlavorData` function can be used to provide the requested data to the Drag Manager. This method is useful when the data needs to be translated by the sender and it would be expensive to compute the data before it is required.

Unlike the functions that add flavors, this function may be called both before and during a drag.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## SetDragSendProc

Sets the send data function for the Drag Manager to use with a particular drag. (Deprecated in Mac OS X v10.5.)

```
OSErr SetDragSendProc (
   DragRef theDrag,
   DragSendDataUPP sendProc,
   void *dragSendRefCon
);
```

**Parameters**

*theDrag*
The drag reference to set the send data function for.

*sendProc*
The send data function that will be called by the Drag Manager when the receiver of a drop requests the flavor data of a flavor that has not been cached by the Drag Manager.

*dragSendRefCon*
A pointer to a reference constant that will be forwarded to your send data function when it is called by the Drag Manager. Use this constant to pass any data you wish to forward to your send data function.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
The Drag Manager caches drag item flavor data when the flavor is added to a drag by calling the `AddDragItemFlavor` function. If `NULL` is passed to the `AddDragItemFlavor` function as the data pointer, the flavor data is not cached and the Drag Manager will call your send data function when the drag item flavor data is requested.

You do not need to provide a send data function if your application never passes `NULL` to the `AddDragItemFlavor` function when adding a drag item flavor to a drag.

Details for how to write a send data function are covered in the "Drag Manager Callbacks" (page 24) section.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## SetDropLocation

Sets the Apple Event descriptor for the drop location of a drag. (Deprecated in Mac OS X v10.5.)

```
OSErr SetDropLocation (
   DragRef theDrag,
   const AEDesc *dropLocation
);
```

### Parameters

*theDrag*

A drag reference.

*dropLocation*

A pointer to the Apple Event descriptor of the drop location to set.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Discussion

This function is typically called by a receive handler before attempting to get any flavor data using the `GetFlavorDataSize` or `GetFlavorData` functions. When a sender application's drag send data function is called to provide flavor data to a receiver, the `GetDropLocation` function can be called to determine the drop location while providing data.

### Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

### Declared In

`Drag.h`

## SetStandardDropLocation

Used by the receiver of a drag to set the standard drop location for a drag. (Deprecated in Mac OS X v10.5.)

```
OSStatus SetStandardDropLocation (
   DragRef theDrag,
   StandardDropLocation dropLocation
);
```

### Parameters

*theDrag*

The drag reference.

*dropLocation*

A value representing the location where the drag was dropped. See "Standard Drop Locations" (page 43) for a description of the values you may use here.

### Return Value

A result code. See "Drag Manager Result Codes" (page 47).

### Availability

Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**
`Drag.h`

## ShowDragHilite

Highlights an area of your window during a drag. (Deprecated in Mac OS X v10.5.)

```
OSErr ShowDragHilite (
   DragRef theDrag,
   RgnHandle hiliteFrame,
   Boolean inside
);
```

**Parameters**

*theDrag*

The drag reference of the drag currently in progress.

*hiliteFrame*

A QuickDraw region of the frame of the window, pane, or shape you wish to highlight, in the window's local coordinate system.

*inside*

Pass `true` to draw the highlighting inside the frame shape. Otherwise it will be drawn outside the frame shape. Note that in either case, the highlight will not include the boundary edge of the frame. This allows you to highlight inside a window frame or a pane, or to highlight outside of a container or object in your window.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
The `ShowDragHilite` function creates a standard drag and drop highlight in your window. Your tracking handler function should call this if a drop is allowed at the current mouse position.

You can only have one highlight showing at a time, and if you call this function when a highlight is currently visible, the first one is removed before the newly requested highlight is shown.

The `ShowDragHilite` function uses a two pixel thick line when drawing the highlight.

The `ShowDragHilite` function assumes that the highlighting should be drawn in the current port. Your application should make sure that the correct port is set before calling the `ShowDragHilite` function. Also, highlighting drawn by the `ShowDragHilite` function is clipped to the current port. Make sure that the port's clip region is appropriately sized to draw the highlighting.

The Drag Manager maintains the currently highlighted portion of your window if you use the `HideDragHilite` and `UpdateDragHilite` functions. If you intend to scroll the window that contains the highlighting, you can use the `DragPreScroll` and `DragPostScroll` functions to properly update the drag highlighting.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## UpdateDragHilite

Updates the portion of the drag highlight that was drawn over by your application. (Deprecated in Mac OS X v10.5.)

```
OSErr UpdateDragHilite (
   DragRef theDrag,
   RgnHandle updateRgn
);
```

**Parameters**

*`theDrag`*
The drag reference.

*`updateRgn`*
The region that needs to be updated, typically the port's `updateRgn`.

**Return Value**
A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**
Use this function if your application draws into the highlighted portion of your window during a drag. For example, dragging over a folder icon in the Finder causes the Finder to redraw the folder icon in its darkened (selected) color. The Finder calls the `UpdateDragHilite` function to redraw any portion of the drag highlight that may have intersected with the folder icon.

You must guarantee, however, that any current highlighting within the `updateRgn` has been completely erased or is clipped out. If this function is asked to highlight over an area which is still highlighted, it will be redrawn incorrectly.

**Availability**
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

**Declared In**
`Drag.h`

## ZoomRects

Animates a rectangle into a second rectangle. (Deprecated in Mac OS X v10.5.)

```
OSErr ZoomRects (
   const Rect *fromRect,
   const Rect *toRect,
   SInt16 zoomSteps,
   ZoomAcceleration acceleration
);
```

**Parameters**

*fromRect*

A pointer to the starting rectangle to animate from, in global coordinates.

*toRect*

A pointer to the ending rectangle to animate to, in global coordinates.

*zoomSteps*

Specifies the number of animation steps to be shown between the source and destination rectangles. The minimum number of steps is 4. If less than 4 steps are specified, 4 will be used. The maximum number of steps is 25. If more than 25 steps are specified, 25 will be used.

*acceleration*

Specifies how the intermediate animation steps will be calculated. Using the `kZoomNoAcceleration` constant makes the distance between steps from the source to the destination equal. Using the `kZoomAccelerate` constant makes each step from the source to the destination increasingly larger, making the animation appear to speed up as it approaches the destination. Using the `kZoomDecelerate` constant makes each step from the source to the destination smaller, making the animation appear to slow down as it approaches the destination.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

The `ZoomRects` function animates a movement between two rectangles on the screen. It does this by drawing gray dithered rectangles incrementally toward the destination rectangle.

The `ZoomRects` function draws on the entire screen, outside of the current port. It does not change any pixels on the screen except during the animation. It also preserves the current port and the port's settings.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

## ZoomRegion

Animates a region's outline from one screen location to another. (Deprecated in Mac OS X v10.5.)

```
OSErr ZoomRegion (
   RgnHandle region,
   Point zoomDistance,
   SInt16 zoomSteps,
   ZoomAcceleration acceleration
);
```

**Parameters**

*region*

The region to animate.

*zoomDistance*

The horizontal and vertical distance from the starting point that the region will animate to.

*zoomSteps*

Specifies the number of animation steps to be shown between the source and destination rectangles. The minimum number of steps is 4. If less than 4 steps are specified, 4 will be used. The maximum number of steps is 25. If more than 25 steps are specified, 25 will be used.

*acceleration*

Specifies how the intermediate animation steps will be calculated. Using the `kZoomNoAcceleration` constant makes the distance between steps from the source to the destination equal. Using the `kZoomAccelerate` constant makes each step from the source to the destination increasingly larger, making the animation appear to speed up as it approaches the destination. Using the `kZoomDecelerate` constant makes each step from the source to the destination smaller, making the animation appear to slow down as it approaches the destination.

**Return Value**

A result code. See "Drag Manager Result Codes" (page 47).

**Discussion**

The `ZoomRegion` function animates a region from one location to another on the screen. It does this by drawing gray dithered regions incrementally toward the destination region.

The `ZoomRegion` function draws on the entire screen, outside of the current port. It does not change any pixels on the screen except during its animation. It also preserves the current port and the port's settings.

**Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`Drag.h`

# Document Revision History

This table describes the changes to *Drag Manager Reference*.

| Date | Notes |
| --- | --- |
| 2006-07-12 | Made minor formatting changes. |
| 2006-07-24 | Added deprecation information. |
| 2003-01-02 | Incorporated documentation for `SetDragImageWithCGImage`, `GetStandardDropLocation`, `SetStandardDropLocation`, `GetDragAllowableActions`, `SetDragAllowableActions`, `GetDragDropAction`, `SetDragDropAction`, and `SetDragImage` functions. |
| | Revised existing Drag Manager result codes and added new ones. |
| | Fixed typographical errors. |

Document Revision History

# Index

## G



## H



## I



## K

## N



## P



## R



## S



## T



## U



## W



## Z