Pasteboard Manager Reference

(Not Recommended)

Developer

Contents

Pasteboard Manager Reference (Not Recommended)

Framework	HIServices/HIServices.h
Companion guide	Pasteboard Manager Programming Guide
Declared in	Pasteboard.h

Important: This document may not represent best practices for current development. Links to downloads and other resources may no longer be valid.

Overview

The Pasteboard Manager lets you create and manipulate pasteboards, which act as holding containers for data to be transferred from one place to another. Typically you use pasteboards to facilitate copy-and-paste or drag-and-drop operations, but you can use them whenever you need to temporarily store data that will be moved elsewhere.

Note: The Pasteboard Manager supersedes the Scrap Manager and the drag flavor APIs in the Drag Manager, adding greater flexibility in the type and quantity of data to be transferred. Pasteboard Manager pasteboards are also fully compatible with Cocoa pasteboards.

Functions by Task

Creating and Using Pasteboards

`PasteboardCreate` (page 8)

Creates a reference to the specified global pasteboard.

`PasteboardSynchronize` (page 15)

Synchronizes the local pasteboard reference to reflect the contents of the global pasteboard.

Manipulating Pasteboard Flavor Data

Functions

PasteboardClear

Clears the contents of the specified pasteboard.


```
OSStatus PasteboardClear (
   PasteboardRef inPasteboard
);
```

Parameters

`inPasteboard`

The pasteboard you want to clear.

Return Value

A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion

After calling this function, the application owns the pasteboard and can add data to it. You must call `PasteboardClear` before modifying a pasteboard.

Availability

Available in OS X v10.3 and later.

Related Sample Code

CarbonSketch

TypeServicesForUnicode

Declared in

`Pasteboard.h`

PasteboardCopyItemFlavorData

Obtains data from a pasteboard for the desired flavor.

```
OSStatus PasteboardCopyItemFlavorData (
   PasteboardRef inPasteboard,
   PasteboardItemID inItem,
   CFStringRef inFlavorType,
   CFDataRef *outData
);
```

Parameters

`inPasteboard`

The pasteboard containing the data.

`inItem`

The identifier for the item whose flavor data you want to obtain.

`inFlavorType`

The flavor of the data you want to obtain, specified as a uniform type identifier.

`outData`

On return, `outData` points to the flavor data. You must release this data using `CFRelease` when you are done using it.

Return Value

A result code. See "Pasteboard Manager Result Codes" (page 22).

Availability

Available in OS X v10.3 and later.

Related Sample Code

CarbonCocoa_PictureCursor
CarbonSketch
Custom_HIView_Tutorial
ImageBrowserView
TypeServicesForUnicode

Declared in

`Pasteboard.h`

PasteboardCopyItemFlavors

Obtains an array of flavors for a specified item in a pasteboard.

```
OSStatus PasteboardCopyItemFlavors (
   PasteboardRef inPasteboard,
   PasteboardItemID inItem,
   CFArrayRef *outFlavorTypes
);
```

Parameters

`inPasteboard`

The pasteboard containing the data.

`inItem`

The identifier for the item whose flavors you want to obtain.

`outFlavorTypes`

On return, `outFlavorTypes` points to an array of flavors, specified as uniform type identifiers. You must release this array by calling `CFRelease` when you are done using it.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Availability
Available in OS X v10.3 and later.

Related Sample Code
CarbonSketch

Custom_HIView_Tutorial

TypeServicesForUnicode

Declared in
`Pasteboard.h`

PasteboardCopyName

Gets the name of a pasteboard.

```
OSStatus PasteboardCopyName (
   PasteboardRef inPasteboard,
   CFStringRef *outName
);
```

Parameters
`inPasteboard`

The pasteboard whose name you want to retrieve.

`outName`

A pointer to a `CFStringRef` variable allocated by the caller. On return, the string contains the name of the specified pasteboard. The caller is responsible for releasing the string.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion
You can use this function to discover the name of a uniquely-named pasteboard so that other processes may access it.

Availability
Available in OS X v10.4 and later.

Declared in
`Pasteboard.h`

PasteboardCopyPasteLocation

Determines the location in which to paste promised data.

```
OSStatus PasteboardCopyPasteLocation (
   PasteboardRef inPasteboard,
   CFURLRef *outPasteLocation
);
```

Parameters

`inPasteboard`

The pasteboard whose paste location you want to determine.

`outPasteLocation`

On return, `outPasteLocation` points to a CFURL indicating the paste location. You must release this URL when you are done with it by calling `CFRelease`.

Return Value

A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion

You typically call this function from your promise keeper callback function to determine where to deliver a promised file. It is also used to tell a translation service where to place a translated file.

This function is analogous to the Drag Manager function `GetDropLocation`.

Availability

Available in OS X v10.3 and later.

Declared in

`Pasteboard.h`

PasteboardCreate

Creates a reference to the specified global pasteboard.

```
OSStatus PasteboardCreate (
   CFStringRef inName,
   PasteboardRef *outPasteboard
);
```


Parameters

`inName`

The name of the pasteboard to reference. To create a new pasteboard, specify a unique name. Pasteboard names should have a reverse DNS-style name to ensure uniqueness. You may also pass one of the following constants:

- `kPasteboardFind` (page 18) to obtain a reference to the global Find pasteboard
- `kPasteboardClipboard` (page 18) to obtain a reference to the global Clipboard
- `kPasteboardUniqueName` (page 19) to ask the Pasteboard Manager to create a new pasteboard with a unique name

`outPasteboard`

A pointer to a `PasteboardRef` variable allocated by the caller. On return, the variable refers to the pasteboard specified in the `inName` parameter.

When you are finished using the pasteboard, you can call `CFRelease` to release the reference. If you do not release the reference, the pasteboard continues to exist even after your application terminates. A subsequently launched application can find a previously created pasteboard by name and examine the data within it.

Return Value

A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion

You can use this function to create a reference to a new or existing pasteboard.

Availability

Available in OS X v10.3 and later.

Related Sample Code

CarbonSketch

TypeServicesForUnicode

Declared in

`Pasteboard.h`

PasteboardGetItemCount

Obtains the number of data items in the specified pasteboard.

```
OSStatus PasteboardGetItemCount (
   PasteboardRef inPasteboard,
```



```
   ItemCount *outItemCount
);
```

Parameters

`inPasteboard`

The pasteboard whose items you want to count.

`outItemCount`

On return, `outItemCount` points to the number of items in the pasteboard.

Return Value

A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion

You usually call this function before calling `PasteboardGetItemIdentifier` (page 11).

Availability

Available in OS X v10.3 and later.

Related Sample Code

CarbonSketch

Custom_HIView_Tutorial

ImageBrowserView

TypeServicesForUnicode

Declared in

`Pasteboard.h`

PasteboardGetItemFlavorFlags

Obtains the flags for a given flavor.

```
OSStatus PasteboardGetItemFlavorFlags (
   PasteboardRef inPasteboard,
   PasteboardItemID inItem,
   CFStringRef inFlavorType,
   PasteboardFlavorFlags *outFlags
);
```

Parameters

`inPasteboard`

The pasteboard containing the data.

`inItem`
The identifier for the item whose flavor flags you want to obtain.

`inFlavorType`
The flavor whose flags you want to obtain.

`outFlags`
On return, `outFlags` points to a bit field containing the flavor flags. See "Pasteboard Flavor Flags" (page 19) for a list of possible values.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Availability
Available in OS X v10.3 and later.

Declared in
`Pasteboard.h`

PasteboardGetItemIdentifier

Obtains the item identifier for an item in a pasteboard.

```
OSStatus PasteboardGetItemIdentifier (
   PasteboardRef inPasteboard,
   CFIndex inIndex,
   PasteboardItemID *outItem
);
```

Parameters
`inPasteboard`
The pasteboard containing the data.

`inIndex`
The one-based index number of the data item whose identifier you want to obtain.

`outItem`
On return, `outItem` points to the item identifier for the data item at index `inIndex`.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion
The item index is one-based to match the convention used by the Drag Manager.

Availability
Available in OS X v10.3 and later.

Related Sample Code
CarbonCocoa_PictureCursor
CarbonSketch
Custom_HIView_Tutorial
ImageBrowserView
TypeServicesForUnicode

Declared in
`Pasteboard.h`

PasteboardPutItemFlavor

Adds flavor data or a promise to the specified pasteboard.

```
OSStatus PasteboardPutItemFlavor (
   PasteboardRef inPasteboard,
   PasteboardItemID inItem,
   CFStringRef inFlavorType,
   CFDataRef inData,
   PasteboardFlavorFlags inFlags
);
```

Parameters
`inPasteboard`
The pasteboard to which to add flavor data or a promise.

`inItem`
The identifier for the item to which to add flavor data or a promise.

`inFlavorType`
The flavor type of the data or promise you are adding, specified as a uniform type identifier.

`inData`
The data to add, or a promise to supply the data later. If you pass a CFData object, you may safely release the object when this function returns. To indicate that the data is promised, pass `kPasteboardPromisedData` (page 21). For more information about promised data, see the Discussion below.

`inFlags`
A bit field of flags for the specified flavor.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion
If you promise data, you must implement a promise keeper callback to deliver the data when asked for it (see `PasteboardPromiseKeeperProcPtr` (page 16) for more details). Typically, you store promises instead of the actual data when the data requires a large overhead to generate. You can call this function multiple times to add multiple flavors to a given item. You should add the flavors in your application's order of preference or richness.

Availability
Available in OS X v10.3 and later.

Related Sample Code
CarbonSketch

TypeServicesForUnicode

Declared in
`Pasteboard.h`

PasteboardResolvePromises

Resolves all promises to a given pasteboard.

```
OSStatus PasteboardResolvePromises (
   PasteboardRef inPasteboard
);
```

Parameters
`inPasteboard`

The pasteboard whose promises your application wants to resolve. If you pass `kPasteboardResolveAllPromises` (page 21), all the promises for all pasteboards handled by the application are resolved.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion
You should call this function when your application quits or otherwise becomes unavailable.

Availability
Available in OS X v10.3 and later.

Declared in
`Pasteboard.h`

PasteboardSetPasteLocation

Sets the paste location before requesting flavor data.

```
OSStatus PasteboardSetPasteLocation (
   PasteboardRef inPasteboard,
   CFURLRef inPasteLocation
);
```

Parameters

`inPasteboard`

The pasteboard you want to obtain flavor data from.

`inPasteLocation`

A Core Foundation URL indicating where to put the data.

Return Value
A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion
Applications that receive pasteboard data in the form of a file should call this function before calling `PasteboardCopyItemFlavorData` (page 5), so the sender knows where to place the promised data. If the requested data was promised by the sender, the sending application calls `PasteboardCopyPasteLocation` (page 8) to determine where to put it.

If a paste location is not applicable for your application, you don't need to call this function.

This function is analogous to the Drag Manager function `SetDropLocation`.

Availability
Available in OS X v10.3 and later.

Declared in
`Pasteboard.h`

PasteboardSetPromiseKeeper

Registers the promise keeper callback function for a pasteboard.


```
OSStatus PasteboardSetPromiseKeeper (
   PasteboardRef inPasteboard,
   PasteboardPromiseKeeperProcPtr inPromiseKeeper,
   void *inContext
);
```

Parameters

`inPasteboard`

The pasteboard to assign the promise keeper callback. If you have multiple pasteboards, you can assign multiple callbacks.

`inPromiseKeeper`

A pointer to your promise keeper callback function. See `PasteboardPromiseKeeperProcPtr` (page 16) for more information about implementing the callback.

`inContext`

A pointer to application-defined data. The value you pass in this parameter is passed to your promise keeper callback function when it is called.

Return Value

A result code. See "Pasteboard Manager Result Codes" (page 22).

Discussion

You must have registered a promise keeper callback function before promising any data on the specified pasteboard.

Availability

Available in OS X v10.3 and later.

Declared in

`Pasteboard.h`

PasteboardSynchronize

Synchronizes the local pasteboard reference to reflect the contents of the global pasteboard.

```
PasteboardSyncFlags PasteboardSynchronize (
   PasteboardRef inPasteboard
);
```

Parameters

`inPasteboard`

The pasteboard you want to synchronize.

Return Value
A flag indicating what synchronization actions occurred.

Discussion
Calling this function compares the local pasteboard reference with its global pasteboard. If the global pasteboard was modified, it updates the local pasteboard reference to reflect this change. You typically call this function whenever your application becomes active, so that its pasteboard information reflects any changes that occurred while it was in the background. This function has low overhead, so you should call it whenever you suspect a global pasteboard may have been changed.

Availability
Available in OS X v10.3 and later.

Related Sample Code
CarbonSketch

TypeServicesForUnicode

Declared in
`Pasteboard.h`

Callbacks

PasteboardPromiseKeeperProcPtr

Defines a pointer to a callback function that supplies the actual data promised on the pasteboard.

```
typedef OSStatus (*PasteboardPromiseKeeperProcPtr)
(
PasteboardRef pasteboard,
PasteboardItemID item,
CFStringRef flavorType,
void* context
);
```

You would declare your promise keeper callback function named `MyPromiseKeeper` like this:

```
OSStatus MyPromiseKeeper (
PasteboardRef pasteboard,
PasteboardItemID item,
CFStringRef flavorType,
void* context
);
```


Parameters

`pasteboard`

The pasteboard whose promise you need to fulfill.

`item`

The pasteboard item identifier containing the promised flavor.

`flavorType`

The flavor of the data being requested, specified as a uniform type identifier.

`context`

The pointer you passed to `PasteboardSetPromiseKeeper` (page 14) in the `inContext` parameter.

Discussion

When promising any flavor data on a pasteboard using `PasteboardPutItemFlavor` (page 12), you must implement a callback function to fulfill that promise.

If your application delivers the promised data as a file, your callback should call `PasteboardCopyPasteLocation` (page 8) to determine where to deliver the requested data and then take the necessary steps to do so.

Availability

Available in OS X v10.3 and later.

Declared in

`Pasteboard.h`

Data Types

PasteboardRef

Defines an opaque type that represents a pasteboard.

```
typedef struct OpaquePasteboardRef *PasteboardRef;
```

Discussion

This is a Core Foundation type. For more information, see *CFType Reference*.

Availability

Available in OS X v10.3 and later.

Declared in

`Pasteboard.h`

PasteboardItemID

Defines a pasteboard item identifier.

```
typedef void* PasteboardItemID;
```

Discussion
You can use any arbitrary (but unique) ID to identify your pasteboard items when placing them on a pasteboard. For example, you may want to identify items by their internal memory address. Only the owning application should interpret its pasteboard item identifiers.

When your application's promise keeper callback function is invoked, it receives a pasteboard item ID, and your application must be able to map that ID to the corresponding promised data.

Availability
Available in OS X v10.3 and later.

Declared in
`Pasteboard.h`

Constants

Pasteboard Name Constants

Define the pasteboard names used when calling `PasteboardCreate` (page 8).

```
#define kPasteboardClipboard CFSTR("com.apple.pasteboard.clipboard")
#define kPasteboardFind CFSTR("com.apple.pasteboard.find")
#define kPasteboardUniqueName (CFStringRef)NULL
```

Constants

`kPasteboardClipboard`

The global Clipboard (used for copy and paste).

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFind`

The global Find pasteboard (used for search fields).

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardUniqueName`

A system-declared pasteboard that is guaranteed to be unique. Each time you call `PasteboardCreate` (page 8) with this constant, you create a different, unique pasteboard.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

Pasteboard Flavor Flags

Indicate useful information associated with pasteboard item flavors.

```
enum {
   kPasteboardFlavorNoFlags = 0,
   kPasteboardFlavorSenderOnly = (1 << 0),
   kPasteboardFlavorSenderTranslated = (1 << 1),
   kPasteboardFlavorNotSaved = (1 << 2),
   kPasteboardFlavorRequestOnly = (1 << 3),
   kPasteboardFlavorSystemTranslated = (1 << 8),
   kPasteboardFlavorPromised = (1 << 9)
};
typedef UInt32 PasteboardFlavorFlags;
```

Constants

`kPasteboardFlavorNoFlags`

No flag information exists for this flavor.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFlavorSenderOnly`

Only the process that added this flavor can see it. Typically used to tag proprietary data that can be cut, dragged, or pasted only within the owning application. This flag is identical to the Drag Manager `flavorSenderOnly` flag.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFlavorSenderTranslated`

The sender translated this data in some fashion before adding it to the pasteboard. The Finder cannot store flavor items marked with this flag in clipping files. This flag is identical to the Drag Manager `flavorSenderTranslated` flag.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFlavorNotSaved`

The receiver should not save the data provided for this flavor. The data may become stale after a drag, or the flavor may be used only to augment another flavor. For example, an application may add a flavor whose only purpose is to supply additional information (say text style) about another flavor. The Finder cannot store flavor items marked with this flag in clipping files. This flag is identical to the Drag Manager `flavorNotSaved` flag.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFlavorRequestOnly`

When the sender sets this flag, this flavor is hidden from calls to `PasteboardCopyItemFlavors` (page 6). However, you can obtain the flavor flags and data by calling `PasteboardGetItemFlavorFlags` (page 10) or `PasteboardCopyItemFlavorData` (page 5). The net result is that applications cannot obtain this flavor unless they are already aware it exists and specifically request it. This functionality can be useful for copying and pasting proprietary data within a suite of applications.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFlavorSystemTranslated`

This data flavor is available through the Translation Manager. That is, the Translation Manager must translate the sender's data before the receiver can receive it. This flag is set automatically when appropriate and cannot be set programmatically. The Finder cannot store flavor items marked with this flag in clipping files. This flag is identical to the Drag Manager `flavorSystemTranslated` flag.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardFlavorPromised`

The data associated with this flavor is not yet on the pasteboard. Typically data is promised to the pasteboard if it requires a lot of overhead to generate. If the receiver requests the data, the sender is notified so it can supply the promised data. This flag is set automatically when appropriate and cannot be set programmatically. This flag is identical to the Drag Manager `flavorDataPromised` flag.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

Pasteboard Synchronization Flags

Indicate the pasteboard status after a call to `PasteboardSynchronize` (page 15).

```
enum {
   kPasteboardModified = (1 << 0),
```



```
   kPasteboardClientIsOwner = (1 << 1)
};
typedef UInt32 PasteboardSyncFlags;
```

Constants

`kPasteboardModified`

The pasteboard was modified since the last time the application accessed it, and the local pasteboard has been synchronized to reflect any changes. Your application should check to see what flavors are now available on the pasteboard and update appropriately (for example, enabling the Paste menu item).

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardClientIsOwner`

The application recently cleared the pasteboard and is its current owner. The application can add flavor data to the pasteboard as desired.

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

Pasteboard Promise Constants

Define constants related to the use of promised data.

```
#define kPasteboardPromisedData (CFDataRef)NULL
#define kPasteboardResolveAllPromises (PasteboardRef)NULL
```

Constants

`kPasteboardPromisedData`

Indicates a promise that pasteboard data will be supplied later. Used when calling `PasteboardPutItemFlavor` (page 12).

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

`kPasteboardResolveAllPromises`

Indicates that all promises on all global pasteboard resources owned by this application should be resolved. Used when calling `PasteboardResolvePromises` (page 13).

Available in OS X v10.3 and later.

Declared in `Pasteboard.h`.

Result Codes

The table below lists the most common error codes returned by the Pasteboard Manager.

Result Code	Value	Description
`badPasteboardSyncErr`	-25130	The pasteboard has been modified and must be synchronized before use. Available in OS X v10.3 and later.
`badPasteboardIndexErr`	-25131	The specified pasteboard item index does not exist. Available in OS X v10.3 and later.
`badPasteboardItemErr`	-25132	The item reference does not exist. Available in OS X v10.3 and later.
`badPasteboardFlavorErr`	-25133	The item flavor does not exist. Available in OS X v10.3 and later.
`duplicatePasteboardFlavorErr`	-25134	The item flavor already exists. Available in OS X v10.3 and later.
`notPasteboardOwnerErr`	-25135	The application did not clear the pasteboard before attempting to add flavor data. Available in OS X v10.3 and later.
`noPasteboardPromiseKeeperErr`	-25136	The application attempted to add promised data without previously registering a promise keeper callback. Available in OS X v10.3 and later.

Document Revision History

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

Date	Notes
2007-06-29	Made minor format and editorial changes.
2004-11-02	Minor corrections.
2004-08-31	First version of this document.

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