Scrap Manager Reference

(Not Recommended)

Carbon > Interapplication Communication

2007-12-11

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

Simultaneously published in the United States and Canada.

Contents

Scrap Manager Reference (Not Recommended)

Framework: Carbon/Carbon.h

Declared in Scrap.h

Overview

Important: The Scrap Manager is deprecated in Mac OS X version 10.5 and later. The replacement API is the Pasteboard Manager. For more information, see *Pasteboard Manager Programming Guide*.

In Mac OS 9 and earlier, the Scrap Manager allowed applications to copy and paste data using the Clipboard. The Scrap Manager was included in Carbon to facilitate the porting of legacy applications to Mac OS X. Only the `LoadScrap` and `UnloadScrap` functions were retained from the original Scrap Manager. However, the Carbon Scrap Manager provided new features, including support for promises.

The Pasteboard Manager supersedes the Scrap Manager and the drag flavor functionality 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.

You should not use the Scrap Manager in new application development.

Functions by Task

Getting Information About the Scrap

`GetScrapFlavorCount` (page 19) Deprecated in Mac OS X v10.5
Obtains a scrap flavor count to use with the `GetScrapFlavorInfoList` function. (Deprecated. Use `PasteboardCopyItemFlavors` instead.)

`GetScrapFlavorFlags` (page 20) Deprecated in Mac OS X v10.5
Obtains information about a specified scrap flavor. (Deprecated. Use `PasteboardGetItemFlavorFlags` instead.)

`GetScrapFlavorInfoList` (page 21) Deprecated in Mac OS X v10.5
Fills an array with items which each describe a corresponding flavor in the scrap. (Deprecated. Use `PasteboardCopyItemFlavors` instead.)

Reading Information From the Scrap

`GetCurrentScrap` (page 17) Deprecated in Mac OS X v10.5
Obtains a reference to the current scrap. (Deprecated. Use `PasteboardCreate` instead.)

`GetScrapByName` (page 18) Deprecated in Mac OS X v10.5
Obtains a reference to a named scrap. (Deprecated. Use `PasteboardCreate` instead.)

`GetScrapFlavorData` (page 19) Deprecated in Mac OS X v10.5
Obtains the data for the specified flavor from the specified scrap. (Deprecated. Use `PasteboardCopyItemFlavorData` instead.)

`GetScrapFlavorSize` (page 21) Deprecated in Mac OS X v10.5
Obtains the size of the data for a specified flavor from a scrap. (Deprecated. Use `PasteboardCopyItemFlavorData` instead.)

Writing Information to the Scrap

`ClearCurrentScrap` (page 15) Deprecated in Mac OS X v10.5
Clears the current scrap. (Deprecated. Use `PasteboardClear` instead.)

`ClearScrap` (page 16) Deprecated in Mac OS X v10.5
Clears a given scrap. (Deprecated. Use `PasteboardClear` instead.)

`PutScrapFlavor` (page 24) Deprecated in Mac OS X v10.5
Puts data on or promises data to the specified scrap. (Deprecated. Use `PasteboardPutItemFlavor` instead.)

Transferring Data Between the Scrap in Memory and the Scrap on Disk

`LoadScrap` (page 23) Deprecated in Mac OS X v10.5
Reads the scrap from the scrap file into memory.

`UnloadScrap` (page 25) Deprecated in Mac OS X v10.5
Writes the scrap from memory to the scrap file.

Working With Scrap Promise Keeper Functions

`CallInScrapPromises` (page 15) Deprecated in Mac OS X v10.5
Forces all promised flavors to be supplied. (Deprecated. Use `PasteboardResolvePromises` instead.)

`DisposeScrapPromiseKeeperUPP` (page 16) Deprecated in Mac OS X v10.5
Disposes of a universal procedure pointer to a function that provides promised scrap data.

`InvokeScrapPromiseKeeperUPP` (page 22) Deprecated in Mac OS X v10.5
Calls a universal procedure pointer to a function that provides promised scrap data.

`NewScrapPromiseKeeperUPP` (page 23) Deprecated in Mac OS X v10.5
Creates a new universal procedure pointer to a function that provides promised scrap data.

`SetScrapPromiseKeeper` (page 25) Deprecated in Mac OS X v10.5
Associates an application-defined promise-keeper function with a scrap or removes an associated promise-keeper. (Deprecated. Use `PasteboardSetPromiseKeeper` instead.)

Callbacks

ScrapPromiseKeeperProcPtr

Defines a pointer to a function the Scrap Manager calls to obtain promised scrap data for a specified flavor.

```
typedef OSStatus (*ScrapPromiseKeeperProcPtr)
(
                ScrapRef scrap,
                ScrapFlavorType flavorType,
                void * userData);
```

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

```
OSStatus MyScrapPromiseKeeperCallback (

            ScrapRef scrap,
            ScrapFlavorType flavorType,
            void * userData);
```

Parameters

scrap

A reference to the scrap to which to supply the promised flavor data.

flavorType

The flavor type to supply the promised data for. Some scrap flavor types are described in "Scrap Flavor Types" (page 9).

userData

An untyped pointer to a buffer, local variable, or other storage location, created and disposed of by your application, and supplied to the Scrap Manager by a previous call to `SetScrapPromiseKeeper` (page 25). For example, this parameter might refer to a pointer or handle to some private scrap data which your promise-keeper function uses in fabricating one or more promised flavors.

This parameter may have a value of `NULL`.

Return Value

A result code. See "Scrap Manager Result Codes" (page 13).

Discussion

To provide a pointer to your promise-keeper function, you create a universal procedure pointer (UPP), using the function `NewScrapPromiseKeeperUPP` (page 23). You can do so with code similar to the following:

```
ScrapPromiseKeeperUPP MyPromiseKeeperUPP;
MyPromiseKeeperUPP = NewScrapPromiseKeeperUPP (&MyScrapPromiseKeeperCallback);
```

You can then associate the UPP with a scrap by passing `MyScrapPromiseKeeperCallback` as a parameter to the `SetScrapPromiseKeeper` (page 25) function.

If a promised flavor is requested through `GetScrapFlavorData` (page 19) or `GetScrapFlavorSize` (page 21), the Scrap Manager calls the application's `ScrapPromiseKeeperProcPtr` function. The scrap promise-keeper function should call `PutScrapFlavor` (page 24) as appropriate to fulfill its promises. Failure to do so—including returning an error or simply neglecting to keep a promise for a flavor—will result in errors being returned to corresponding callers of `GetScrapFlavorData` or `GetScrapFlavorSize`.

Under Mac OS X, the scrap promise-keeper function can be called during any call to `GetScrapFlavorData` or `GetScrapFlavorSize`. Under Mac OS 8 and 9, the Carbon Scrap Manager still must support non-Carbon callers of `GetScrap`, which does not know about promised flavors. As a result, the Carbon Scrap Manager must make sure all promises have been kept when the application is suspended.

It is okay to keep a promise without ever receiving a call to a scrap promise-keeper function. You should not call `WaitNextEvent` or any similar function from your promise-keeper function because the promise-keeper might be running at suspend event time.

After you are finished with a promise-keeper function, and have removed it by passing `NULL` to the `SetScrapPromiseKeeper` (page 25) function, you can dispose of the UPP with the `DisposeScrapPromiseKeeperUPP` (page 16) function. However, don't dispose of the UPP if any other scrap uses it or if you plan to use it again.

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

Declared In
`Scrap.h`

Data Types

ScrapFlavorInfo

Describes a single flavor within a scrap.

```
struct ScrapFlavorInfo {
    ScrapFlavorType flavorType;
    ScrapFlavorFlags flavorFlags;
};
typedef struct ScrapFlavorInfo ScrapFlavorInfo;
```

Fields

`flavorType`
The flavor type for the flavor. Some values for flavor types are described in "Scrap Flavor Types" (page 9).

`flavorFlags`
The flavor flags for the flavor. Some values for flavor flags are described in "Scrap Flavor Flags" (page 10).

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

Declared In
`Scrap.h`

ScrapRef

Defines a reference to a scrap.

```
typedef struct OpaqueScrapRef * ScrapRef;
```

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

Declared In
`Scrap.h`

ScrapPromiseKeeperUPP

Defines a universal procedure pointer to a scrap promise–keeper function.

```
typedef ScrapPromiseKeeperProcPtr ScrapPromiseKeeperUPP;
```

Discussion
See `ScrapPromiseKeeperProcPtr` (page 7) for more information on scrap promise–keeper functions.

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

Declared In
`Scrap.h`

ScrapFlavorType

Defines a scrap flavor type.

```
typedef FourCharCode ScrapFlavorType;
```

Discussion
Some values for scrap flavor types are shown in "Scrap Flavor Types" (page 9).

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

Declared In
`Scrap.h`

Constants

Scrap Flavor Types

Identify commonly used scrap flavor types.

```
enum {
    kScrapFlavorTypePicture = 'PICT',
    kScrapFlavorTypeText = 'TEXT',
    kScrapFlavorTypeTextStyle = 'styl',
    kScrapFlavorTypeMovie = 'moov',
    kScrapFlavorTypeSound = 'snd ',
    kScrapFlavorTypeUnicode = 'utxt',
    kScrapFlavorTypeUnicodeStyle = 'ustl'
};
```

Constants

`kScrapFlavorTypePicture`
Specifies the contents of a `PicHandle`.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorTypeText`
Specifies a stream of text characters.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorTypeTextStyle`
Specifies a style scrap structure. For more information, see `TEGetStyleScrapHandle` in *Inside Mac OS X: TextEdit Reference*.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorTypeMovie`
Specifies a reference to a QuickTime movie.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorTypeSound`
Specifies the contents of a `'snd '` resource. For more information, see the functions `SndRecord` and `SndPlay` in *Inside Mac OS X: Sound Manager Reference*.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorTypeUnicode`
Specifies a stream of 16-bit Unicode characters.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorTypeUnicodeStyle`
Specifies Unicode style information; defined by ATSUI and used by Textension.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

Scrap Flavor Flags

Describe attributes of a scrap flavor.

```
enum {
    kScrapFlavorMaskNone = 0x00000000,
    kScrapFlavorMaskSenderOnly = 0x00000001,
    kScrapFlavorMaskTranslated = 0x00000002
};
typedef UInt32 ScrapFlavorFlags;
```

Constants

`kScrapFlavorMaskNone`

No flavors in the scrap.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorMaskSenderOnly`

The flavor is intended to be visible to the sender only. This is typically used to save a private flavor into the scrap so that other public promised flavors can be derived from it on demand. If another process put a flavor with this flag on the scrap, your process will never see the flavor, so you do not need to test for this flag.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

`kScrapFlavorMaskTranslated`

The flavor has been translated (or will be translated when the promise is kept) from some other flavor in the scrap, either by the sender or by the Translation Manager.

Available in Mac OS X v10.0 and later.

Declared in `Scrap.h`.

Discussion

To determine the attributes of a scrap flavor, call the `GetScrapFlavorFlags` (page 20) function.

Reserved Flavor Type

Identifies a flavor type reserved by the Scrap Manager.

```
enum {
    kScrapReservedFlavorType = 'srft'
};
```

Discussion

`kScrapReservedFlavorType` is a flavor type which is reserved for use by the Scrap Manager. If you pass it to the Scrap Manager, it will be rejected.

Unknown Flavor Data Size Constant

Indicates that the size of the scrap flavor data is unknown.

```
enum {
    kScrapFlavorSizeUnknown = -1
};
```

Discussion

When promising a scrap flavor, it is okay if you don't yet know how big the flavor data will be. In this case, pass `kScrapFlavorSizeUnknown` for the flavor data size.

Options for the GetScrapByName Function

Define options passed to the `GetScrapByName` function.

```
enum {
    kScrapGetNamedScrap = 0,
    kScrapClearNamedScrap = 0x00000001
};
```

Constants

`kScrapGetNamedScrap`

Get current named scrap without bumping or clearing the scrap.

Available in Mac OS X v10.1 and later.

Declared in `Scrap.h`.

`kScrapClearNamedScrap`

Acquire the named scrap, bumping and clearing the scrap.

Available in Mac OS X v10.1 and later.

Declared in `Scrap.h`.

Invalid Scrap Reference

Defines an invalid scrap reference.

```
#define kScrapRefNone ((ScrapRef)NULL)
```

Discussion

`kScrapRefNone` is guaranteed to be an invalid `ScrapRef`. This is convenient when initializing application variables.

Named Scraps

Specify Apple–defined scrap names for the `GetScrapByName` function.

```
#define kScrapClipboardScrap CFSTR("com.apple.scrap.clipboard")
#define kScrapFindScrap CFSTR("com.apple.scrap.find")
```

Constants

`kScrapClipboardScrap`

Traditional clipboard scrap.

Available in Mac OS X v10.1 and later.

Declared in `Scrap.h`.

`kScrapFindScrap`

Compatible with Cocoa's global find scrap.

Available in Mac OS X v10.1 and later.

Declared in `Scrap.h`.

Result Codes

The most common result codes returned by Scrap Manager are listed below.

Result Code	Value	Description
`needClearScrapErr`	-100	Scrap does not exist (same as `noScrapErr`) Available in Mac OS X v10.0 and later.
`noScrapErr`	-100	Scrap does not exist (not initialized) Available in Mac OS X v10.0 and later.
`noTypeErr`	-102	No data of the requested format type in scrap Available in Mac OS X v10.0 and later.
`scrapFlavorNotFoundErr`	-102	No data of the requested format type in scrap (same as `noTypeErr`) Available in Mac OS X v10.0 and later.
`internalScrapErr`	-4988	Scrap Manager has encountered an internal error Available in Mac OS X v10.0 and later.
`duplicateScrapFlavorErr`	-4989	Data (or a promise for data) of the specified format already exists in the scrap Available in Mac OS X v10.0 and later.
`badScrapRefErr`	-4990	Invalid scrap reference Available in Mac OS X v10.0 and later.
`processStateIncorrectErr`	-4991	Available in Mac OS X v10.0 and later.
`scrapPromiseNotKeptErr`	-4992	No data supplied for the promised flavor Available in Mac OS X v10.0 and later.
`noScrapPromiseKeeperErr`	-4993	No scrap promise–keeper function set for the scrap Available in Mac OS X v10.0 and later.
`nilScrapFlavorDataErr`	-4994	Available in Mac OS X v10.0 and later.
`scrapFlavorFlagsMismatchErr`	-4995	Available in Mac OS X v10.0 and later.
`scrapFlavorSizeMismatchErr`	-4996	Available in Mac OS X v10.0 and later.
`illegalScrapFlavorFlagsErr`	-4997	Invalid scrap flavor flags Available in Mac OS X v10.0 and later.
`illegalScrapFlavorTypeErr`	-4998	Invalid scrap flavor Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`illegalScrapFlavorSizeErr`	-4999	Invalid size for scrap flavor data Available in Mac OS X v10.0 and later.

Gestalt Constants

You can check for version and feature availability information by using the Scrap Manager selectors defined in the Gestalt Manager. For more information see *Gestalt Manager Reference*.

Deprecated Scrap Manager Reference (Not Recommended) Functions

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

Deprecated in Mac OS X v10.5

CallInScrapPromises

Forces all promised flavors to be supplied. (Deprecated in Mac OS X v10.5. Use `PasteboardResolvePromises` instead.)

```
OSStatus CallInScrapPromises (
   void
);
```

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
Before quitting, your application should call `CallInScrapPromises` in order to ensure the user's ability to paste into other applications. Your application should call `CallInScrapPromises` even if your application does not explicitly promise any flavors.

It doesn't hurt to call `CallInScrapPromises` more than once, though promise-keeper functions may be asked to keep promises they already tried and failed.

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
`Scrap.h`

ClearCurrentScrap

Clears the current scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardClear` instead.)

```
OSStatus ClearCurrentScrap (
   void
);
```

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
Call `ClearCurrentScrap` immediately when the user requests a Copy or Cut operation, even if you maintain a private scrap. You should not wait until receiving a suspend event to call `ClearCurrentScrap`. You don't need to put any data on the scrap immediately, although it's perfectly fine to do so. You do need to call `GetCurrentScrap` (page 17) after `ClearCurrentScrap` so you'll have a valid scrap reference to pass to other functions.

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
`Scrap.h`

ClearScrap

Clears a given scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardClear` instead.)

```
OSStatus ClearScrap (
   ScrapRef *inOutScrap
);
```

Parameters

inOutScrap
A pointer to a scrap reference. On input, this parameter should refer to the scrap to clear. On output, `ClearScrap` returns a reference to the cleared scrap.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
`ClearScrap` will clear the scrap passed in and return the incremented `ScrapRef` value. `ClearScrap` behaves similarly to the `GetScrapByName` function when called with the `kScrapClearNamedScrap` option, with the benefit of not requiring a name in the event one is not available.

Availability
Available in Mac OS X v10.1 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Scrap.h`

DisposeScrapPromiseKeeperUPP

Disposes of a universal procedure pointer to a function that provides promised scrap data. (Deprecated in Mac OS X v10.5.)

```
void DisposeScrapPromiseKeeperUPP (
   ScrapPromiseKeeperUPP userUPP
);
```

Parameters

userUPP

The UPP to dispose of.

Discussion

See `ScrapPromiseKeeperProcPtr` (page 7) for more information on promise keeper functions.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Declared In

`Scrap.h`

GetCurrentScrap

Obtains a reference to the current scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardCreate` instead.)

```
OSStatus GetCurrentScrap (
   ScrapRef *scrap
);
```

Parameters

scrap

A pointer to a scrap reference. On return, this scrap reference refers to the current scrap.

Return Value

A result code. See "Scrap Manager Result Codes" (page 13).

Discussion

Your application can determine if the scrap contents have changed by storing the scrap reference returned by `GetCurrentScrap` and comparing it against the scrap reference returned by `GetCurrentScrap` at a later time. If the two scrap references are different, the scrap has changed.

Carbon applications should use `GetCurrentScrap` instead of checking the `convertClipboardFlag` in the `EventRecord`.

The `ScrapRef` obtained via `GetCurrentScrap` becomes invalid and unusable after the scrap is cleared. That is, the scrap reference is valid until a Carbon client calls `ClearCurrentScrap` (page 15), a Classic client calls `ZeroScrap` , or a Cocoa client calls `declareTypes:owner:`.

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

`Scrap.h`

GetScrapByName

Obtains a reference to a named scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardCreate` instead.)

```
OSStatus GetScrapByName (
   CFStringRef name,
   OptionBits options,
   ScrapRef *scrap
);
```

Parameters

name

A CFString containing the name of the scrap to obtain. You may specify a standard scrap by passing one of the constants described in "Named Scraps" (page 12) in this parameter.

options

A value indicating whether the specified scrap should be cleared after the reference is returned. See "Options for the GetScrapByName Function" (page 12) for more information.

scrap

A pointer to a scrap reference. On return, this scrap reference refers to the scrap named in the `name` parameter.

Return Value

A result code. See "Scrap Manager Result Codes" (page 13).

Discussion

`GetScrapByName` allows access to an indefinite number of public or private scraps. The constant `kScrapClipboardScrap` refers to the "current" scrap. `kScrapFindScrap` allows Carbon applications to interact seamlessly with Cocoa's global find scrap. Note that calling:

```
GetScrapByName( kScrapClipboardScrap, kScrapGetNamedScrap, &scrap  );
```

is an exact match to the call:

```
GetCurrentScrap( &scrap );
```

Additionally, a call to:

```
 GetScrapByName( kScrapClipboardScrap, kScrapClearNamedScrap, &scrap  );
```

is a replacement for the sequence:

```
ClearCurrentScrap();
GetCurrentScrap( &scrap );
```

You can use this API to generate your own private scraps to use as a high level interprocess communication between your main and helper applications. Apple recommends using the Java naming convention for your scraps, for example, com.mycompany.scrap.secret.

Availability

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Scrap.h`

GetScrapFlavorCount

Obtains a scrap flavor count to use with the `GetScrapFlavorInfoList` function. (Deprecated in Mac OS X v10.5. Use `PasteboardCopyItemFlavors` instead.)

```
OSStatus GetScrapFlavorCount (
   ScrapRef scrap,
   UInt32 *infoCount
);
```

Parameters

scrap

A reference to the scrap to get the flavor count for.

infoCount

A pointer to a count variable. On return, specifies the flavor count for the specified scrap.

Return Value

A result code. See "Scrap Manager Result Codes" (page 13).

Discussion

For related information, see `GetScrapFlavorInfoList` (page 21).

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

`Scrap.h`

GetScrapFlavorData

Obtains the data for the specified flavor from the specified scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardCopyItemFlavorData` instead.)

```
OSStatus GetScrapFlavorData (
   ScrapRef scrap,
   ScrapFlavorType flavorType,
   Size *byteCount,
   void *destination
);
```

Parameters

scrap

A reference to the scrap to get data from.

flavorType

The flavor type to obtain data for. Some flavor types are described in "Scrap Flavor Types" (page 9).

byteCount

A pointer to a size variable. Before calling `GetScrapFlavorData`, specify the maximum number of bytes your buffer can contain. On return, `GetScrapFlavorData` provides the number of bytes of data that were actually available, even if this is more than you requested.

destination

A pointer to a buffer, local variable, or other storage location created and disposed of by your application. The size, in bytes, must be at least as large as the value you pass in the `byteCount` parameter. On return, this buffer contains the specified flavor data. The amount of data returned will not exceed the value you passed in `byteCount`, even if the number of bytes of available data is more than you specified in `byteCount`.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
This function blocks until the specified flavor data is available.

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
`Scrap.h`

GetScrapFlavorFlags

Obtains information about a specified scrap flavor. (Deprecated in Mac OS X v10.5. Use `PasteboardGetItemFlavorFlags` instead.)

```
OSStatus GetScrapFlavorFlags (
   ScrapRef scrap,
   ScrapFlavorType flavorType,
   ScrapFlavorFlags *flavorFlags
);
```

Parameters

scrap

A reference to the scrap to check.

flavorType

The flavor type to check for. Some scrap flavor types are described in "Scrap Flavor Types" (page 9).

flavorFlags

A pointer to a variable of type `ScrapFlavorFlags`; values for this type are described in "Scrap Flavor Flags" (page 10). On return, this variable contains information about the flavor specified by the `flavorType` parameter.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
The `GetScrapFlavorFlags` function tells you whether the scrap contains data for a particular flavor and, if it does, provides some information about that flavor. This function never blocks, and is useful for deciding whether to enable the Paste item in your Edit menu, among other things.

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
`Scrap.h`

GetScrapFlavorInfoList

Fills an array with items which each describe a corresponding flavor in the scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardCopyItemFlavors` instead.)

```
OSStatus GetScrapFlavorInfoList (
   ScrapRef scrap,
   UInt32 *infoCount,
   ScrapFlavorInfo info[]
);
```

Parameters

scrap

A reference to the scrap to get the flavor information for.

infoCount

A pointer to a count variable. Before calling `GetScrapFlavorInfoList`, set the value to the number of flavors to get information for. Your application typically obtains the flavor count by calling `GetScrapFlavorCount` (page 19).

On return, specifies the number of flavors for which information was supplied, which may be smaller than the number requested.

info

An array of type `ScrapFlavorInfo` (page 8), whose size is indicated by the `infoCount` parameter. The array is created and disposed of by your application. On return, the array elements contain the flavor information.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

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
`Scrap.h`

GetScrapFlavorSize

Obtains the size of the data for a specified flavor from a scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardCopyItemFlavorData` instead.)

```
OSStatus GetScrapFlavorSize (
   ScrapRef scrap,
   ScrapFlavorType flavorType,
   Size *byteCount
);
```

Parameters

scrap

A reference to the scrap to get the flavor data size from.

flavorType

The flavor type to obtain the size for. Some flavor types are described in "Scrap Flavor Types" (page 9).

byteCount

A pointer to a size variable. On return, this variable contains the byte count for the data of the specified flavor.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
This function will block until the size of the data is available. This may mean blocking until the data itself is available, since some scrap senders don't know how big a flavor will be until they've made the flavor data. `GetScrapFlavorSize` is intended as a prelude to allocating memory and calling `GetScrapFlavorData` (page 19).

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
`Scrap.h`

InvokeScrapPromiseKeeperUPP

Calls a universal procedure pointer to a function that provides promised scrap data. (Deprecated in Mac OS X v10.5.)

```
OSStatus InvokeScrapPromiseKeeperUPP (
   ScrapRef scrap,
   ScrapFlavorType flavorType,
   void *userData,
   ScrapPromiseKeeperUPP userUPP
);
```

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
You should not need to use the function `InvokeScrapPromiseKeeperUPP`, as the system calls your scrap promise keeper function for you.

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

Deprecated in Mac OS X v10.5.

Declared In
`Scrap.h`

LoadScrap

Reads the scrap from the scrap file into memory. (Deprecated in Mac OS X v10.5.)

```
OSStatus LoadScrap (
   void
);
```

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
The function allocates memory in your application's heap to hold the scrap before reading the scrap into memory. The scrap file is located in the System Folder of the startup volume and has the filename as indicated by the `scrapName` field of the scrap information structure (usually "Clipboard"). If the scrap is already in memory, this function does nothing.

Special Considerations

In Mac OS X, this function does nothing and is no longer 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
`Scrap.h`

NewScrapPromiseKeeperUPP

Creates a new universal procedure pointer to a function that provides promised scrap data. (Deprecated in Mac OS X v10.5.)

```
ScrapPromiseKeeperUPP NewScrapPromiseKeeperUPP (
   ScrapPromiseKeeperProcPtr userRoutine
);
```

Parameters

`userRoutine`

A pointer to your scrap promise keeper function.

Return Value
A UPP to the scrap promise keeper function. See the description of the `ScrapPromiseKeeperUPP` data type.

Discussion
See `ScrapPromiseKeeperProcPtr` (page 7) for more information on promise keeper functions.

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

Deprecated in Mac OS X v10.5.

Declared In
`Scrap.h`

PutScrapFlavor

Puts data on or promises data to the specified scrap. (Deprecated in Mac OS X v10.5. Use `PasteboardPutItemFlavor` instead.)

```
OSStatus PutScrapFlavor (
   ScrapRef scrap,
   ScrapFlavorType flavorType,
   ScrapFlavorFlags flavorFlags,
   Size flavorSize,
   const void *flavorData
);
```

Parameters

scrap

A reference to the scrap to supply data or promises to.

flavorType

The flavor type to supply or promise the data for. Some flavor types are described in "Scrap Flavor Types" (page 9).

flavorFlags

A variable of type `ScrapFlavorFlags` that you use to supply information about the flavor specified by the `flavorType` parameter. See "Scrap Flavor Flags" (page 10) for a description of the values you can use in this parameter.

flavorSize

The size of the data you are supplying or promising, in bytes. If you don't know the size, pass `kScrapFlavorSizeUnknown` to place a promise for data of undetermined size on the scrap. If you pass 0 in this parameter, a flavor with no expected data—not a promise—is placed on the scrap, and the value of the *flavorData* parameter is ignored.

flavorData

A pointer to a buffer, local variable, or other storage location, created and disposed of by your application. Before calling `PutScrapFlavor` to put flavor data on the scrap, store the data in this buffer. For information on the number of bytes of data, see the description of the `flavorSize` parameter.

Pass `NULL` for this parameter to indicate you will provide data through a subsequent call to `PutScrapFlavor`, either later in the same code flow or during execution of your `ScrapPromiseKeeperProcPtr` (page 7) callback.

The last time you can provide scrap flavor data is when your scrap promise-keeper function gets called. It is not possible to call `PutScrapFlavor` while handling a suspend event; suspend events under Carbon don't work the way they do under Mac OS 8 and 9.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
`PutScrapFlavor` is different than `PutScrap` in that it includes a `ScrapRef` parameter and it supports promising a flavor for later delivery, rather than supplying it immediately.

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
`Scrap.h`

SetScrapPromiseKeeper

Associates an application-defined promise-keeper function with a scrap or removes an associated promise-keeper. (Deprecated in Mac OS X v10.5. Use `PasteboardSetPromiseKeeper` instead.)

```
OSStatus SetScrapPromiseKeeper (
   ScrapRef scrap,
   ScrapPromiseKeeperUPP upp,
   const void *userData
);
```

Parameters

scrap

A reference to the scrap to set the promise-keeper function for.

upp

A universal procedure pointer to a scrap promise-keeper function. For more information, see `ScrapPromiseKeeperProcPtr` (page 7)

You can remove a promise-keeper function from a scrap by passing `NULL` for this parameter.

userData

An untyped pointer to a buffer, local variable, or other storage location, created and disposed of by your application. This value is passed to the promise-keeper function specified by the `upp` parameter, which can do whatever it needs to do with the value. For example, you might pass a pointer or handle to some private scrap data that your promise-keeper function uses in fabricating one or more promised flavors.

If your promise-keeper function has no need for special user data, pass `NULL` for this parameter.

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

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
`Scrap.h`

UnloadScrap

Writes the scrap from memory to the scrap file. (Deprecated in Mac OS X v10.5.)

```
OSStatus UnloadScrap (
   void
);
```

Return Value
A result code. See "Scrap Manager Result Codes" (page 13).

Discussion
This function releases the memory occupied by the scrap in your application's heap. The scrap file is located in the System Folder of the startup volume and has the filename as indicated by the `scrapName` field of the scrap information structure (usually "Clipboard"). If the scrap is already on the disk, this function does nothing.

Special Considerations

In Mac OS X, this function does nothing and is no longer 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
`Scrap.h`

Document Revision History

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

Date	Notes
2007-12-11	Added deprecation information.
2003-01-02	Added documentation for `ClearScrap` and `GetScrapByName` functions.
	Fixed description of `PutScrapFlavor` behavior.
	Revised existing Scrap Manager result codes and added new ones.
	Fixed typographical errors.

Document Revision History

Index

O

P

R

S

U