NSPasteboard

Inherits From:	NSObject
Conforms To:	NSObject (NSObject)
Declared In:	AppKit/NSPasteboard.h

Class at a Glance

Purpose
An NSPasteboard object is an interface to a pasteboard server that allows you to transfer data between applications, as in copy, cut, and paste operations. The data can be placed in the pasteboard server in a variety of representations.

Principal Attributes
- Owners
- Data types
- Change count
- Name

Creation
+ generalPasteboard
+ pasteboardWithName:

Commonly Used Methods

– types	Returns an NSArray of pasteboard data types.
– declareTypes:owner:	Prepares NSPasteboard to receive new data.
– dataForType:	Reads data from a pasteboard.
– setData:forType:	Writes data to a pasteboard.
– stringForType:	Reads an NSString from a pasteboard.
– setStringForType:	Writes an NSString to a pasteboard.

Class Description

NSPasteboard objects transfer data to and from the pasteboard server. The server is shared by all running applications. It contains data that the user has cut or copied, as well as other data that one application wants to transfer to another. NSPasteboard objects are an application's sole interface to the server and to all pasteboard operations.

Named Pasteboards

Data in the pasteboard server is associated with a name that indicates how it's to be used. Each set of data and its associated name is, in effect, a separate pasteboard, distinct from the others. An application keeps a separate NSPasteboard object for each named pasteboard that it uses. There are five standard pasteboards in common use:

General pasteboard	The pasteboard that's used for ordinary cut, copy, and paste operations. It holds the contents of the last selection that's been cut or copied.
Font pasteboard	The pasteboard that holds font and character information and supports Copy Font and Paste Font commands that my be implemented in a text editor.
Ruler pasteboard	The pasteboard that holds information about paragraph formats in support of the Copy Ruler and Paste Ruler commands that may be implemented in a text editor.
Find pasteboard	The pasteboard that holds information about the current state of the active application's Find panel. This information permits users to enter a search string into the Find panel, then switch to another application to conduct another search.
Drag pasteboard	The pasteboard that stores data to be moved as the result of a drag operation.

Each standard pasteboard is identified by a unique name (stored in global string objects):

NSGeneralPboard
NSFontPboard
NSRulerPboard
NSFindPboard
NSDragPboard

You can create private pasteboards by asking for an NSPasteboard object with any name other than those listed above. Data in a private pasteboard may then be shared by passing its name between applications.

The NSPasteboard class makes sure there's never more than one object for each named pasteboard on the computer. If you ask for a new object when one has already been created for the pasteboard with that name, the existing object will be returned.

Data Types

Data can be placed in the pasteboard server in more than one representation. For example, an image might be provided both in Tag Image File Format (TIFF) and as encapsulated PostScript code (EPS). Multiple representations give pasting applications the option of choosing which data type to use. In general, an application taking data from the pasteboard should choose the richest representation it can handle—rich text over plain ASCII, for example. An application putting data in the pasteboard should promise to supply it in as many data types as possible, so that as many different applications as possible can use it.

Filtering services transform the data from one representation to another. Typically, these services aren't invoked until data is read from a pasteboard.

Data types are identified by NSString objects containing the full type name. These global variables identify the string objects for the standard pasteboard types:

Type	**Description**
NSColorPboardType	NSColor data
NSDataLinkPboardType	Defines a link between documents
NSFileContentsPboardType	A representation of a file's contents
NSFilenamesPboardType	NSString designating one or more file names
NSFontPboardType	Font and character information
NSPostScriptPboardType	Encapsulated PostScript code (EPS)
NSRulerPboardType	Paragraph formatting information
NSRTFPboardType	Rich Text Format (RTF)
NSRTFDPboardType	RTFD formatted file contents
NSSelectionPboardType	Describes a selection for use with data linking
NSStringPboardType	NSString data
NSTabularTextPboardType	NSString containing tab-separated fields of text
NSTIFFPboardType	Tag Image File Format (TIFF)

Typically, data is written to the pasteboard using **setData:forType:** and read using **dataForType:**. Some of these types can only be written with certain methods. For instance, NSFilenamesPboardType's form is an array of NSStrings and requires special handling. Use these methods to write these types:

Type	**Writing Method**	**Reading Method**
NSColorPboardType	NSColor class methods	NSColor class methods
NSFileContentsPboardType	**writeFileContents:**	**readFileContentsType:toFile:**
NSFilenamesPboardType	**setPropertyList:forType:**	**propertyListForType:**
NSStringPboardType	**setString:forType:**	**stringForType:**

You don't have to write the data (using **setData:forType:**) in all types that you've declared for the pasteboard: This avoids unneeded conversions. If data is requested from a pasteboard in a format that's not present, the owner of the pasteboard receives a **pasteboard:provideDataForType:** message notifying it that it needs to supply the data in that format. It then supplies data in the requested type by invoking one of the **setData:forType:**, **setString:forType:**, or **setPropertyList:forType:** methods on the pasteboard.

The class methods **pasteboardByFilteringData:ofType:**, **pasteboardByFilteringFile:**, and **pasteboardByFilteringTypesInPasteboard:** return a pasteboard with data that is filtered into all types derivable from the current types using available filter services. (For more information on filter services see **/NextLibrary/Documentation/NextDev/TasksAndConcepts/ProgrammingTopics/Services.rtf**.) The pasteboards returned by these methods are autoreleased instances of NSPasteboard.

Types other than those listed above can also be used. For example, your application may keep data in a private format that's richer than any of the existing types. That format can also be used as a pasteboard type.

Reading and Writing RTFD Data

The NSRTFDPboardType is used for the contents of an RTFD file package (a directory containing an RTF text file and one or many EPS and TIFF image files). There are several ways to work with RTFD data. If you have an NSFileWrapper object that represents an RTFD file wrapper, you can send it the **serializedRepresentation** method to return the RTFD data and write that to the pasteboard as follows:

```
NSFileWrapper *tempRTFDData = [[[NSFileWrapper alloc]
    initWithPath:@"/tmp/foo.rtfd"] autorelease];
[pboard setData:[tempRTFDData serializedRepresentation]
    forType:NSRTFDPboardType];
```

In addition to NSFileWrapper, classes such as NSAttributedString and NSText can return RTFD data. If you're using one of these classes, you would do the following to write RTFD data to the pasteboard:

```
NSAttributedString *attrString;
...
[pboard setData:[attrString RTFDFromRange:NSMakeRange(0, [attrString length])]
    forType:NSRTFDPboardType];
```

Change Count

The change count is a computer-wide variable that increments every time the contents of the pasteboard changes (a new owner is declared). An independent change count is maintained for each named pasteboard.

By examining the change count, an application can determine whether the current data in the pasteboard is the same as the data it last received.

The **changeCount**, **addTypes:owner:**, and **declareTypes:owner:** methods return the change count. A **types** or **availableTypeFromArray:** message should be sent by the pasteboard before reading data so that the change count is valid.

Errors

Except where errors are specifically mentioned in the method descriptions, any communications error with the pasteboard server raises an NSPasteboardCommunicationException.

Method Types

Creating and releasing an NSPasteboard object	
	+ generalPasteboard
	+ pasteboardByFilteringData:ofType:
	+ pasteboardByFilteringFile:
	+ pasteboardByFilteringTypesInPasteboard:
	+ pasteboardWithName:
	+ pasteboardWithUniqueName
	+ typesFilterableTo:
	– releaseGlobally
Referring to a pasteboard by name	– name
Writing data	– addTypes:owner:
	– declareTypes:owner:
	– setData:forType:
	– setPropertyList:forType:
	– setString:forType:
	– writeFileContents:
Determining Types	– availableTypeFromArray:
	– types
Reading Data	– changeCount
	– dataForType:
	– propertyListForType:
	– readFileContentsType:toFile:
	– stringForType:
Methods Implemented by the Owner	
	– pasteboardChangedOwner:
	– pasteboard:provideDataForType:

Class Methods

generalPasteboard

+ (NSPasteboard *)**generalPasteboard**

Returns the general NSPasteboard. This invokes **pasteboardWithName:** to obtain the pasteboard.

pasteboardByFilteringData:ofType:

+ (NSPasteboard *)**pasteboardByFilteringData:**(NSData *)*data* **ofType:**(NSString *)*type*

Creates and returns a new pasteboard with a unique name that supplies *data* in as many types as possible given the available filter services. The returned pasteboard also declares data of the supplied type *type*.

No filter service is invoked until the data is actually requested, so invoking this method is reasonably inexpensive.

pasteboardByFilteringFile:

+ (NSPasteboard *)**pasteboardByFilteringFile:**(NSString *)*filename*

Creates and returns a new pasteboard with a unique name that supplies the data in *filename* in as many types as possible given the available filter services. No filter service is invoked until the data is actually requested, so invoking this method is reasonably inexpensive.

pasteboardByFilteringTypesInPasteboard:

+ (NSPasteboard *)**pasteboardByFilteringTypesInPasteboard:**(NSPasteboard *)*pasteboard*

Creates and returns a new pasteboard with a unique name that supplies the data on *pasteboard* in as many types as possible given the available filter services. This process can be thought of as expanding the pasteboard, since the new pasteboard generally will contain more representations of the data than *pasteboard*.

This method returns *pasteboard* if *pasteboard* was returned by one of the **pasteboardByFiltering...** methods, so a pasteboard can’t be expanded multiple times. This method only returns the original types and the types that can be created as a result of a single filter; the pasteboard will not have defined types that are the result of translation by multiple filters.

No filter service is invoked until the data is actually requested, so invoking this method is reasonably inexpensive.

pasteboardWithName:

+ (NSPasteboard *)**pasteboardWithName:**(NSString *)*name*

Returns the pasteboard for the *name* pasteboard. A new object is created only if the application doesn't yet have a NSPasteboard object for the specified name; otherwise, the existing one is returned. To get a standard pasteboard, *name* should be one of the following variables:

NSGeneralPboard
NSFontPboard
NSRulerPboard
NSFindPboard
NSDragPboard

Other names can be assigned to create private pasteboards for other purposes.

pasteboardWithUniqueName

+ (NSPasteboard *)**pasteboardWithUniqueName**

Creates and returns a new pasteboard with a name that is guaranteed to be unique with respect to other pasteboards on the computer. This method is useful for applications that implement their own interprocess communication using pasteboards.

typesFilterableTo:

+ (NSArray *)**typesFilterableTo:**(NSString *)*type*

Returns an autoreleased array listing the types of data that *type* can be converted to by available filter services. The array contains the original type.

Instance Methods

addTypes:owner:

– (int)**addTypes:**(NSArray *)*newTypes* **owner:**(id)*newOwner*

Adds the data types in *newTypes* to the NSPasteboard and declares a new owner *newOwner*. This method can be useful when multiple entities (such as a combination of application and library methods) contribute data for a single copy command. It should only be invoked after a **declareTypes:owner:** message has been sent for the same types. The owner for the new types may be different from the owner(s) of the previously declared types.

Returns the new change count, or 0 in case of an error.

See also: – changeCount

availableTypeFromArray:

– (NSString *)**availableTypeFromArray:**(NSArray *)*types*

Scans the types defined by *types* and returns the first type that matches a type declared on the receiving NSPasteboard.

A **types** or **availableTypeFromArray:** message should be sent before reading any data from the NSPasteboard.

changeCount

– (int)**changeCount**

Returns the NSPasteboard’s change count.

See also: **– declareTypes:owner:**

dataForType:

– (NSData *)**dataForType:**(NSString *)*dataType*

Reads the *dataType* representation of the current contents of the NSPasteboard. *dataType* should be one of the types returned by the **types** method. A **types** or **availableTypeFromArray:** message should be sent before invoking **dataForType:**.

If the data is successfully read, this method returns the data. It returns **nil** if the contents of the pasteboard have changed (if the change count has been incremented by a **declareTypes:owner:** message) since they were last checked with the **types** method. It also returns **nil** if the pasteboard server can’t supply the data in time—for example, if the NSPasteboard’s owner is slow in responding to a **pasteboard:provideDataForType:** message and the interprocess communication times out. All other errors raise an NSPasteboardCommunicationException exception.

If **nil** is returned, the application should put up a panel informing the user that it was unable to carry out the paste operation.

The NSData object that this method returns is autoreleased.

declareTypes:owner:

– (int)**declareTypes:**(NSArray *)*newTypes* **owner:**(id)*newOwner*

Prepares the NSPasteboard for a change in its contents by declaring the new types of data it will contain and a new owner. This is the first step in responding to a user’s copy or cut command and must precede the messages that actually write the data. A **declareTypes:owner:** message essentially changes the contents of the pasteboard: It invalidates the current contents of the pasteboard and increments its change count.

newTypes is an array of NSStrings that name types the new contents of the pasteboard may assume. The types should be ordered according to the preference of the source application, with the most preferred type coming first (typically, the richest representation).

The *newOwner* is the object responsible for writing data to the pasteboard in all the types listed in *newTypes*. You can write the data immediately after declaring the types, or wait until it's required for a paste operation. If you wait, the owner will receive a **pasteboard:provideDataForType:** message requesting the data in a particular type when it's needed. You might choose to write data immediately for the most preferred type, but wait for the others to see whether they'll be requested.

The *newOwner* can be NULL if data is provided for all types immediately. Otherwise, the owner should be an object that won't be released. It should not, for example, be the NSView that displays the data if that NSView is in a window that might be closed.

Returns the pasteboard's new change count.

See also: – **setString:forType:**, – **addTypes:owner:**, – **changeCount**

name

– (NSString *)**name**

Returns the NSPasteboard's name.

See also: + pasteboardWithName:

propertyListForType:

– (id)**propertyListForType:**(NSString *)*dataType*

Returns a *property list* object using the type specified by *dataType*.

A property list is an object of the NSArray, NSData, NSDictionary, or NSString classes—or any combination thereof.

A **types** or **availableTypeFromArray:** message should be sent before invoking **propertyListForType:**.

This method invokes **dataForType:**.

See also: – setPropertyList:forType:

readFileContentsType:toFile:

– (NSString *)**readFileContentsType:**(NSString *)*type* **toFile:**(NSString *)*filename*

Reads data representing a file's contents from the NSPasteboard, and writes it to the file *filename*. An **availableTypeFromArray:** or **types** message should be sent before invoking **readFileContentsType:toFile:**.

Data of any file contents *type* should only be read using this method. *type* should generally be specified; if *type* is NULL, a type based on filename's extension (as returned by **NSCreateFileContentsPboardType()**) is substituted. If data matching *type* isn't found on the NSPasteboard, data of type NSFileContentsPboardType is requested. Returns the name of the file that the data was actually written to.

See also: – writeFileContents:

releaseGlobally

– (void)**releaseGlobally**

Releases the NSPasteboard's resources in the pasteboard server. Since an NSPasteboard object is an autoreleased instance of NSPasteboard, it isn't released by this method, and its retain count isn't changed.

After this method is invoked, no other application will be able to use the named NSPasteboard. A temporary, privately named pasteboard can be released this way when it's no longer needed, but a standard NSPasteboard should never be released globally.

setData:forType:

– (BOOL)**setData:**(NSData *)*data* **forType:**(NSString *)*dataType*

Writes data to the pasteboard server. *dataType* gives the type of data being written; it must be a type that was declared in the previous **declareTypes:owner:** message. *data* points to the data to be sent to the pasteboard server.

Returns YES if the data is successfully written or returns NO if ownership of the pasteboard has changed. Any other error raises an NSPasteboardCommunicationException.

See also: **– setPropertyList:forType:**, **– setString:forType:**

setPropertyList:forType:

– (BOOL)**setPropertyList:**(id)*propertyList* **forType:**(NSString *)*dataType*

Writes data to the pasteboard server. *dataType* gives the type of data being written; it must be a type that was declared in the previous **declareTypes:owner:** message. *propertyList* points to the data to be sent to the pasteboard server.

This method invokes **setData:forType:** with a serialized property list parameter.

Returns YES if the data is successfully written or returns NO if ownership of the pasteboard has changed. Any other error raises an NSPasteboardCommunicationException.

See also: – setString:forType:

setString:forType:

– (BOOL)**setString:**(NSString *)*string* **forType:**(NSString *)*dataType*

Writes data to the pasteboard server. *dataType* gives the type of data being written; it must be a type that was declared in the previous **declareTypes:owner:** message. *string* points to the data to be sent to the pasteboard server.

This method invokes **setPropertyList:forType:** to perform the write.

Returns YES if the data is successfully written or returns NO if ownership of the pasteboard has changed. Any other error raises an NSPasteboardCommunicationException.

See also: – **setData:forType:**, – **setString:forType:**

stringForType:

– (NSString *)**stringForType:**(NSString *)*dataType*

Returns an NSString using the type specified by *dataType*. A **types** or **availableTypeFromArray:** message should be sent before invoking **stringForType:**.

This method invokes **propertyListForType:** to obtain the string.

types

– (NSArray *)**types**

Returns an array of the NSPasteboard's data types.

Returns an array of the types that were declared for the current contents of the NSPasteboard. The array is an array of NSStrings holding the type names. Types are listed in the same order that they were declared.

A **types** or **availableTypeFromArray:** message should be sent before reading any data from the NSPasteboard.

See also: **– declareTypes:owner:**, **– dataForType:**, **NSUniqueString()**

writeFileContents:

– (BOOL)**writeFileContents:**(NSString *)*filename*

Writes the contents of the file *filename* to the NSPasteboard object and declares the data to be of type NSFileContentsPboardType and also of a type appropriate for the file's extension (as returned by **NSCreateFileContentsPboardType()** when passed the files extension), if it has one. Returns YES if the data from *filename* was successfully written to the NSPasteboard and NO otherwise.

See also: – readFileContentsType:toFile:

Methods Implemented by the Owner

pasteboardChangedOwner:

– (void)**pasteboardChangedOwner:**(NSPasteboard *)*sender*

Notifies a prior owner of the *sender* pasteboard (and owners of representations on the pasteboard) that the pasteboard has changed owners. This method is optional and need only be implemented by pasteboard owners that need to know when they have lost ownership. The owner is not able to read the contents of the pasteboard when responding to this method. The owner should be prepared to receive this method at any time, even from within the **declareTypes:owner:** used to declare ownership.

See also: – changeCount

pasteboard:provideDataForType:

– (void)**pasteboard:**(NSPasteboard *)*sender*
provideDataForType:(NSString *)*type*

Implemented by the owner (previously declared in a **declareTypes:owner:** message) to provide promised data. The owner receives a **pasteboard:provideDataForType:** message from the sender pasteboard when the data is required for a paste operation; *type* gives the type of data being requested. The requested data should be written to *sender* using the **setData:forType:**, **setPropertyList:forType:**, or **setString:forType:** methods.

pasteboard:provideDataForType: messages may also be sent to the owner when the application is shut down through Application's **terminate:** method. This is the method that's invoked in response to a Quit command. Thus the user can copy something to the pasteboard, quit the application, and still paste the data that was copied.

A **pasteboard:provideDataForType:** message is sent only if *type* data hasn't already been supplied. Instead of writing all data types when the cut or copy operation is done, an application can choose to implement this method to provide the data for certain types only when they're requested.

If an application writes data to the NSPasteboard in the richest, and therefore most preferred, type at the time of a cut or copy operation, its **pasteboard:provideDataForType:** method can simply read that data from the pasteboard, convert it to the requested *type*, and write it back to the pasteboard as the new type.