
NSFileWrapper Class Reference

Data Management: File Management



2009-05-26



Apple Inc.
© 2009 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Cocoa, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Finder is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSFileWrapper Class Reference 5

Overview	5
Adopted Protocols	6
Tasks	6
Creating File Wrappers	6
Querying File Wrappers	6
Accessing File-Wrapper Information	7
Updating File Wrappers	7
Serializing	8
Accessing Files	8
Writing Files	8
Instance Methods	8
addFileWrapper:	8
addRegularFileWithContents:preferredFilename:	9
fileAttributes	10
filename	11
fileWrappers	11
icon	12
initWithDirectoryWithFileWrappers:	13
initWithRegularFileWithContents:	13
initWithSymbolicLinkWithDestinationURL:	14
initWithSerializedRepresentation:	15
initWithURL:options:error:	15
isDirectory	16
isRegularFile	16
isSymbolicLink	17
keyForFileWrapper:	17
matchesContentsOfURL:	18
preferredFilename	19
readFromURL:options:error:	19
regularFileContents	20
removeFileWrapper:	20
serializedRepresentation	21
setFileAttributes:	21
setFilename:	22
setIcon:	22
setPreferredFilename:	23
symbolicLinkDestinationURL	24
writeToURL:options:originalContentsURL:error:	24
Constants	25
File Wrapper Reading Options	25

File Wrapper Writing Options 27

Appendix A **Deprecated NSFileWrapper Methods 29**

Deprecated in Mac OS X v10.6 29

addFilePath: 29

addSymbolicLinkWithDestination:preferredFilename: 30

initSymbolicLinkWithDestination: 30

initWithPath: 31

needsToBeUpdatedFromPath: 32

symbolicLinkDestination 32

updateFromPath: 33

writeToFile:atomically:updateFileNames: 33

Document Revision History 35

Index 37

NSFileWrapper Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Application File Management
Declared in	NSFileWrapper.h
Related sample code	DictionaryController File Wrappers with Core Data Documents FunHouse Quartz Composer WWDC 2005 TextEdit SimpleStickies

Overview

The `NSFileWrapper` class provides access to the attributes and contents of file-system nodes. A **file-system node** is a file, directory, or symbolic link. Instances of this class are known as **file wrappers**.

File wrappers represent a file-system node as an object that can be displayed as an image (and possibly edited in place), saved to the file system, or transmitted to another application. It can also store an icon for representing the node in a document or in a dragging operation.

There are three types of file wrappers:

- **Regular-file file wrapper:** Represents a regular file.
- **Directory file wrapper:** Represents a directory.
- **Symbolic-link file wrapper:** Represents a symbolic link.

A file wrapper has these attributes:

- **Filename.** Name of the file-system node the file wrapper represents.
- **Icon:** Image that represents the file wrapper to the user.
- **file-system attributes.** See *NSFileManager Class Reference* for information on the contents of the *attributes* dictionary.

- **Regular-file contents.** Applicable only to regular-file file wrappers.
- **File wrappers.** Applicable only to directory file wrappers.
- **Destination node.** Applicable only to symbolic-link file wrappers.

Adopted Protocols

NSCoding

```
encodeWithCoder:
initWithCoder:
```

Tasks

Creating File Wrappers

This class has several designated initializers.

- [initWithURL:options:error:](#) (page 15)
Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the URL.
- [initWithDirectoryWithFileWrappers:](#) (page 13)
Initializes the receiver as a directory file wrapper, with a given file-wrapper list.
- [initWithRegularFileWithContents:](#) (page 13)
Initializes the receiver as a regular-file file wrapper.
- [initWithSymbolicLinkWithDestinationURL:](#) (page 14)
Initializes the receiver as a symbolic-link file wrapper that links to a specified file.
- [initWithSerializedRepresentation:](#) (page 15)
Initializes the receiver as a regular-file file wrapper from given serialized data.
- [initWithSymbolicLinkWithDestination:](#) (page 30) **Deprecated in Mac OS X v10.6**
Initializes the receiver as a symbolic-link file wrapper. (**Deprecated.** Use [initWithSymbolicLinkWithDestinationURL:](#) (page 14) instead.)
- [initWithPath:](#) (page 31) **Deprecated in Mac OS X v10.6**
Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the path. (**Deprecated.** Use [initWithURL:options:error:](#) (page 15) instead.)

Querying File Wrappers

- [isRegularFile](#) (page 16)
Indicates whether the receiver is a regular-file file wrapper.
- [isDirectory](#) (page 16)
Indicates whether the receiver is a directory file wrapper.

- [isSymbolicLink](#) (page 17)
Indicates whether the receiver is a symbolic-link file wrapper.

Accessing File-Wrapper Information

- [fileWrappers](#) (page 11)
Returns the file wrappers contained by a directory file wrapper.
- [addFileWrapper:](#) (page 8)
Adds a child file wrapper to the receiver, which must be a directory file wrapper.
- [removeFileWrapper:](#) (page 20)
Removes a child file wrapper from the receiver, which must be a directory file wrapper.
- [addRegularFileWithContents:preferredFilename:](#) (page 9)
Creates a regular-file file wrapper with the given contents and adds it to the receiver, which must be a directory file wrapper.
- [keyForFileWrapper:](#) (page 17)
Returns the dictionary key used by a directory to identify a given file wrapper.
- [symbolicLinkDestinationURL](#) (page 24)
Provides the URL referenced by the receiver, which must be a symbolic-link file wrapper.
- [addFileWithPath:](#) (page 29) **Deprecated in Mac OS X v10.6**
Creates a file wrapper from a given file-system node and adds it to the receiver, which must be a directory file wrapper. (**Deprecated.** Use [addFileWrapper:](#) (page 8) instead.)
- [addSymbolicLinkWithDestination:preferredFilename:](#) (page 30) **Deprecated in Mac OS X v10.6**
Creates a symbolic-link file wrapper pointing to a given file-system node and adds it to the receiver, which must be a directory file wrapper. (**Deprecated.** Use [addFileWrapper:](#) (page 8) instead.)
- [symbolicLinkDestination](#) (page 32) **Deprecated in Mac OS X v10.6**
Provides the pathname referenced by the receiver, which must be a symbolic-link file wrapper. (**Deprecated.** Use [symbolicLinkDestinationURL](#) (page 24) instead.)

Updating File Wrappers

- [matchesContentsOfURL:](#) (page 18)
Indicates whether the contents of a file wrapper matches a directory, regular file, or symbolic link on disk.
- [readFromURL:options:error:](#) (page 19)
Recursively rereads the entire contents of a file wrapper from the specified location on disk.
- [needsToBeUpdatedFromPath:](#) (page 32) **Deprecated in Mac OS X v10.6**
Indicates whether the file wrapper needs to be updated to match a given file-system node. (**Deprecated.** Use [matchesContentsOfURL:](#) (page 18) instead.)
- [updateFromPath:](#) (page 33) **Deprecated in Mac OS X v10.6**
Updates the file wrapper to match a given file-system node. (**Deprecated.** Use [readFromURL:options:error:](#) (page 19) instead.)

Serializing

- [serializedRepresentation](#) (page 21)
Returns the contents of the file wrapper as an opaque collection of data.

Accessing Files

- [filename](#) (page 11)
Provides the filename of a file wrapper.
- [setFilename:](#) (page 22)
Specifies the filename of a file wrapper.
- [preferredFilename](#) (page 19)
Provides the preferred filename for a file wrapper.
- [setPreferredFilename:](#) (page 23)
Specifies the receiver's preferred filename.
- [icon](#) (page 12)
Returns an icon that represents the file wrapper to the user.
- [setIcon:](#) (page 22)
Specifies an image that can be used to represent the file wrapper to the user.
- [fileAttributes](#) (page 10)
Returns a file wrapper's file attributes.
- [setFileAttributes:](#) (page 21)
Specifies a file wrapper's file attributes.
- [regularFileContents](#) (page 20)
Returns the contents of the file-system node associated with a regular-file file wrapper.

Writing Files

- [writeToURL:options:originalContentsURL:error:](#) (page 24)
Recursively writes the entire contents of a file wrapper to a given file-system URL.
- [writeToFile:atomically:updateFileNames:](#) (page 33) **Deprecated in Mac OS X v10.6**
Writes a file wrapper's contents to a given file-system node. (**Deprecated**. Use [writeToURL:options:originalContentsURL:error:](#) (page 24) instead.)

Instance Methods

addFileWrapper:

Adds a child file wrapper to the receiver, which must be a directory file wrapper.

- (NSString *)addFileWrapper:(NSFileWrapper *)*child*

Parameters*child*

File wrapper to add to the directory.

Return Value

Dictionary key used to store *fileWrapper* in the directory's list of file wrappers. The dictionary key is a unique filename, which is the same as the passed-in file wrapper's preferred filename unless that name is already in use as a key in the directory's dictionary of children. See *Working With Directory Wrappers in Application File Management* for more information about the file-wrapper list structure.

Discussion

Use this method to add an existing file wrapper as a child of a directory file wrapper. If the file wrapper does not have a preferred filename, use the [setPreferredFilename:](#) (page 23) method to give it one before calling `addFileWrapper:`. To create a new file wrapper and add it to a directory, use the [addRegularFileWithContents:preferredFilename:](#) (page 9) method.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

This method raises `NSInvalidArgumentException` if the child file wrapper doesn't have a preferred name.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [addRegularFileWithContents:preferredFilename:](#) (page 9)
- [removeFileWrapper:](#) (page 20)
- [fileWrappers](#) (page 11)
- [preferredFilename](#) (page 19)

Related Sample Code

File Wrappers with Core Data Documents

Declared In

`NSFileWrapper.h`

addRegularFileWithContents:preferredFilename:

Creates a regular-file file wrapper with the given contents and adds it to the receiver, which must be a directory file wrapper.

```
- (NSString *)addRegularFileWithContents:(NSData *)data preferredFilename:(NSString *)filename
```

Parameters*data*

Contents for the new regular-file file wrapper.

filename

Preferred filename for the new regular-file file wrapper.

Return Value

Dictionary key used to store the new file wrapper in the directory's list of file wrappers. The dictionary key is a unique filename, which is the same as the passed-in file wrapper's preferred filename unless that name is already in use as a key in the directory's dictionary of children. See *Working With Directory Wrappers in Application File Management* for more information about the file-wrapper list structure.

Discussion

This is a convenience method. The default implementation allocates a new file wrapper, initializes it with [initWithRegularFileWithContents:](#) (page 13), sends it [setPreferredFilename:](#) (page 23), adds it to the directory with [addFileWrapper:](#) (page 8), and returns what [addFileWrapper:](#) returned.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `filename`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [addFileWrapper:](#) (page 8)
- [removeFileWrapper:](#) (page 20)
- [fileWrappers](#) (page 11)

Related Sample Code

FunHouse
SimpleStickies

Declared In

`NSFileWrapper.h`

fileAttributes

Returns a file wrapper's file attributes.

- (NSDictionary *)fileAttributes

Return Value

File attributes, in a dictionary of the same sort as that returned by `attributesOfItemAtPath:error:` (`NSFileManager`).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setFileAttributes:](#) (page 21)

Declared In

`NSFileWrapper.h`

filename

Provides the filename of a file wrapper.

- (NSString *)filename

Return Value

The file wrapper's filename; `nil` when the file wrapper has no corresponding file-system node.

Discussion

The filename is used for record-keeping purposes only and is set automatically when the file wrapper is created from the file system using [initWithURL:options:error:](#) (page 15) and when it's saved to the file system using [writeToURL:options:originalContentsURL:error:](#) (page 24) (although this method allows you to request that the filename not be updated).

The filename is usually the same as the preferred filename, but might instead be a name derived from the preferred filename. You can use this method to get the name of a child that's just been read. Don't use this method to get the name of a child that's about to be written, because the name might be about to change; send [keyForFileWrapper:](#) (page 17) to the parent instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [preferredFilename](#) (page 19)
- [setFilename:](#) (page 22)

Related Sample Code

File Wrappers with Core Data Documents
Quartz Composer WWDC 2005 TextEdit

Declared In

NSFileWrapper.h

fileWrappers

Returns the file wrappers contained by a directory file wrapper.

- (NSDictionary *)fileWrappers

Return Value

A key-value dictionary of the file wrappers contained in the directory. The dictionary contains entries whose values are the file wrappers and whose keys are the unique filenames that have been assigned to each one. See [Working With Directory Wrappers](#) in *Application File Management* for more information about the file-wrapper list structure.

Discussion

Returns a dictionary whose values are the file wrapper's children and whose keys are the unique filenames that have been assigned to each one. This method may return `nil` if the user modifies the directory after you call [readFromURL:options:error:](#) (page 19) or [initWithURL:options:error:](#) (page 15) but before `NSFileWrapper` has read the contents of the directory. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [filename](#) (page 11)
- [addFileWrapper:](#) (page 8)

Related Sample Code

File Wrappers with Core Data Documents

FunHouse

Declared In

`NSFileWrapper.h`

icon

Returns an icon that represents the file wrapper to the user.

```
- (NSImage *)icon
```

Return Value

Icon that represents the file wrapper; `nil` when the file wrapper has no icon.

Discussion

You don't have to use this icon; for example, a file viewer typically looks up icons automatically based on file extensions, and so wouldn't need this one. Similarly, if a file wrapper represents an image file, you can display the image directly rather than a file icon.

This method may return `nil` if the file wrapper is a child created when its parent was read from the file system, and the child was modified before it was read. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Because the `NSImage` object that's returned might be shared by many `NSFileWrapper` objects, you must not mutate it. If you need to mutate the returned object, make a copy first and mutate the copy instead.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setIcon:](#) (page 22)
- [initWithURL:options:error:](#) (page 15)
- [readFromURL:options:error:](#) (page 19)

Related Sample Code

DictionaryController

Declared In

`NSFileWrapper.h`

initWithDirectoryWithFileWrappers:

Initializes the receiver as a directory file wrapper, with a given file-wrapper list.

```
- (id)initWithDirectoryWithFileWrappers:(NSDictionary *)childrenByPreferredName
```

Parameters

childrenByPreferredName

Key-value dictionary of file wrappers with which to initialize the receiver. The dictionary must contain entries whose values are the file wrappers that are to become children and whose keys are filenames. See *Working With Directory Wrappers* in *Application File Management* for more information about the file-wrapper list structure.

Return Value

Initialized file wrapper for *fileWrappers*.

Discussion

After initialization, the file wrapper is not associated with a file-system node until you save it using [writeToURL:options:originalContentsURL:error:](#) (page 24).

The receiver is initialized with open permissions: anyone can read, write, or modify the directory on disk.

If any file wrapper in the directory doesn't have a preferred filename, its preferred name is automatically set to its corresponding key in the *childrenByPreferredName* dictionary.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [setFileAttributes:](#) (page 21)

Related Sample Code

File Wrappers with Core Data Documents

FunHouse

SimpleStickies

Declared In

NSFileWrapper.h

initWithRegularFileWithContents:

Initializes the receiver as a regular-file file wrapper.

```
- (id)initWithRegularFileWithContents:(NSData *)contents
```

Parameters

contents

Contents of the file.

Return Value

Initialized regular-file file wrapper containing *contents*.

Discussion

After initialization, the file wrapper is not associated with a file-system node until you save it using [writeToURL:options:originalContentsURL:error:](#) (page 24).

The file wrapper is initialized with open permissions: anyone can write to or read the file wrapper. .

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)
- [regularFileContents](#) (page 20)

Related Sample Code

File Wrappers with Core Data Documents
FunHouse

Declared In

NSFileWrapper.h

initWithSymbolicLinkWithDestinationURL:

Initializes the receiver as a symbolic-link file wrapper that links to a specified file.

```
- (id)initWithSymbolicLinkWithDestinationURL:(NSURL *)url
```

Parameters

url

URL of the file the file wrapper is to reference.

Return Value

Initialized symbolic-link file wrapper referencing *url*.

Discussion

The file wrapper is not associated with a file-system node until you save it using [writeToURL:options:originalContentsURL:error:](#) (page 24).

The file wrapper is initialized with open permissions: anyone can modify or read the file reference. .

Availability

Available in Mac OS X v10.6 and later.

See Also

- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)

Declared In

NSFileWrapper.h

initWithSerializedRepresentation:

Initializes the receiver as a regular-file file wrapper from given serialized data.

```
- (id)initWithSerializedRepresentation:(NSData *)serializedRepresentation
```

Parameters

serializedRepresentation

Serialized representation of a file wrapper in the format used for the `NSFileContentsPboardType` pasteboard type. Data of this format is returned by such methods as [serializedRepresentation](#) (page 21) and `RTFDFromRange:documentAttributes:` (`NSAttributedString`).

Return Value

Regular-file file wrapper initialized from *serializedRepresentation*.

Discussion

The file wrapper is not associated with a file-system node until you save it using [writeToURL:options:originalContentsURL:error:](#) (page 24).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)

Declared In

`NSFileWrapper.h`

initWithURL:options:error:

Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the URL.

```
- (id)initWithURL:(NSURL *)url options:(NSFileWrapperReadingOptions)options
  error:(NSError **)outError
```

Parameters

url

URL of the file-system node the file wrapper is to represent.

options

Option flags for reading the node located at *url*. See [“File Wrapper Reading Options”](#) (page 25) for possible values.

outError

If an error occurs, upon return contains an `NSError` object that describes the problem. Pass `NULL` if you do not want error information.

Return Value

File wrapper for the file-system node at *url*. May be a directory, file, or symbolic link, depending on what is located at the URL. Returns `NO` (0) if reading is not successful.

Discussion

If *url* is a directory, this method recursively creates file wrappers for each node within that directory. Use the [fileWrappers](#) (page 11) method to get the file wrappers of the nodes contained by the directory.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [fileWrappers](#) (page 11)
- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)
- [readFromURL:options:error:](#) (page 19)

Declared In

NSFileWrapper.h

isDirectory

Indicates whether the receiver is a directory file wrapper.

- (BOOL)isDirectory

Return Value

YES when the receiver is a directory file wrapper, NO otherwise.

Discussion

Invocations of [readFromURL:options:error:](#) (page 19) may change what is returned by subsequent invocations of this method if the type of the file on disk has changed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [isRegularFile](#) (page 16)
- [isSymbolicLink](#) (page 17)

Declared In

NSFileWrapper.h

isRegularFile

Indicates whether the receiver is a regular-file file wrapper.

- (BOOL)isRegularFile

Return Value

YES when the receiver is a regular-file wrapper, NO otherwise.

Discussion

Invocations of [readFromURL:options:error:](#) (page 19) may change what is returned by subsequent invocations of this method if the type of the file on disk has changed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [isDirectory](#) (page 16)
- [isSymbolicLink](#) (page 17)

Declared In

NSFileWrapper.h

isSymbolicLink

Indicates whether the receiver is a symbolic-link file wrapper.

- (BOOL)isSymbolicLink

Return Value

YES when the receiver is a symbolic-link file wrapper, NO otherwise.

Discussion

Invocations of [readFromURL:options:error:](#) (page 19) may change what is returned by subsequent invocations of this method if the type of the file on disk has changed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [isDirectory](#) (page 16)
- [isRegularFile](#) (page 16)

Declared In

NSFileWrapper.h

keyForFileWrapper:

Returns the dictionary key used by a directory to identify a given file wrapper.

- (NSString *)keyForFileWrapper:(NSFileWrapper *)child

Parameters

child

The child file wrapper for which you want the key.

Return Value

Dictionary key used to store the file wrapper in the directory's list of file wrappers. The dictionary key is a unique filename, which may not be the same as the passed-in file wrapper's preferred filename if more than one file wrapper in the directory's dictionary of children has the same preferred filename. See *Working With Directory Wrappers* in *Application File Management* for more information about the file-wrapper list structure. Returns `nil` if the file wrapper specified in `child` is not a child of the directory.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [filename](#) (page 11)

Declared In

NSFileWrapper.h

matchesContentsOfURL:

Indicates whether the contents of a file wrapper matches a directory, regular file, or symbolic link on disk.

- (BOOL)matchesContentsOfURL:(NSURL *)url

Parameters

url

URL of the file-system node with which to compare the file wrapper.

Return Value

YES when the contents of the file wrapper match the contents of *url*, NO otherwise.

Discussion

The contents of files are not compared; matching of regular files is based on file modification dates. For a directory, children are compared against the files in the directory, recursively.

Because children of directory file wrappers are not read immediately by the [initWithURL:options:error:](#) (page 15) method unless the `NSFileWrapperReadingImmediate` reading option is used, even a newly-created directory file wrapper might not have the same contents as the directory on disk. You can use this method to determine whether the file wrapper's contents in memory need to be updated.

If the file wrapper needs updating, use the [readFromURL:options:error:](#) (page 19) method with the `NSFileWrapperReadingImmediate` reading option.

This table describes which attributes of the file wrapper and file-system node are compared to determine whether the file wrapper matches the node on disk:

File-wrapper type	Comparison determinants
Regular file	Modification date and access permissions.
Directory	Children (recursive).
Symbolic link	Destination pathname.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [readFromURL:options:error:](#) (page 19)
- [fileAttributes](#) (page 10)

Declared In

NSFileWrapper.h

preferredFilename

Provides the preferred filename for a file wrapper.

- (NSString *)preferredFilename

Return Value

The file wrapper's preferred filename.

Discussion

This name is normally used as the dictionary key when a child file wrapper is added to a directory file wrapper. However, if another file wrapper with the same preferred name already exists in the directory file wrapper when the receiver is added, the filename assigned as the dictionary key may differ from the preferred filename.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [filename](#) (page 11)
- [setPreferredFilename:](#) (page 23)

Declared In

NSFileWrapper.h

readFromURL:options:error:

Recursively rereads the entire contents of a file wrapper from the specified location on disk.

- (BOOL)readFromURL:(NSURL *)url options:(NSFileWrapperReadingOptions)options error:(NSError **)outError

Parameters*url*

URL of the file-system node corresponding to the file wrapper.

*options*Option flags for reading the node located at *url*. See “[File Wrapper Reading Options](#)” (page 25) for possible values.*outError*If an error occurs, upon return contains an `NSError` object that describes the problem. Pass `NULL` if you do not want error information.**Return Value**YES if successful. If not successful, returns NO after setting *outError* to an `NSError` object that describes the reason why the file wrapper could not be reread.**Discussion**

When reading a directory, children are added and removed as necessary to match the file system.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [initWithURL:options:error:](#) (page 15)
- [fileWrappers](#) (page 11)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)
- [writeToURL:options:originalContentsURL:error:](#) (page 24)

Declared In

NSFileWrapper.h

regularFileContents

Returns the contents of the file-system node associated with a regular-file file wrapper.

- (NSData *)regularFileContents

Return Value

Contents of the file-system node the file wrapper represents.

Discussion

This method may return `nil` if the user modifies the file after you call [readFromURL:options:error:](#) (page 19) or [initWithURL:options:error:](#) (page 15) but before `NSFileWrapper` has read the contents of the file. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a regular-file file wrapper.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithRegularFileWithContents:](#) (page 13)
- [readFromURL:options:error:](#) (page 19)

Related Sample Code

File Wrappers with Core Data Documents

FunHouse

SimpleStickies

Declared In

NSFileWrapper.h

removeFileWrapper:

Removes a child file wrapper from the receiver, which must be a directory file wrapper.

- (void)removeFileWrapper:(NSFileWrapper *)child

Parameters

child

File wrapper to remove from the directory.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [addFileWrapper:](#) (page 8)
- [addRegularFileWithContents:preferredFilename:](#) (page 9)
- [fileWrappers](#) (page 11)

Related Sample Code

File Wrappers with Core Data Documents

Declared In

`NSFileWrapper.h`

serializedRepresentation

Returns the contents of the file wrapper as an opaque collection of data.

```
- (NSData *)serializedRepresentation
```

Return Value

The file wrapper's contents in the format used for the pasteboard type `NSFileContentsPboardType`.

Discussion

Returns an `NSData` object suitable for passing to [initWithSerializedRepresentation:](#) (page 15). This method may return `nil` if the user modifies the contents of the file-system node after you call [readFromURL:options:error:](#) (page 19) or [initWithURL:options:error:](#) (page 15) but before `NSFileWrapper` has read the contents of the file. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithSerializedRepresentation:](#) (page 15)

Declared In

`NSFileWrapper.h`

setFileAttributes:

Specifies a file wrapper's file attributes.

```
- (void)setFileAttributes:(NSDictionary *)fileAttributes
```

Parameters

fileAttributes

File attributes for the file wrapper, in a dictionary of the same sort as that used by `setAttributes:ofItemAtPath:error:` (`NSFileManager`).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [fileAttributes](#) (page 10)
- [writeToURL:options:originalContentsURL:error:](#) (page 24)

Declared In

NSFileWrapper.h

setFilename:

Specifies the filename of a file wrapper.

```
- (void)setFilename:(NSString *)filename
```

Parameters

filename

Filename of the file wrapper.

Discussion

The file name is a dictionary key used to store *fileWrapper* in a directory's list of child file wrappers. The dictionary key is a unique filename, which is the same as the child file wrapper's preferred filename unless that name is already in use as a key in the directory's dictionary of children. See *Working With Directory Wrappers* in *Application File Management* for more information about the file-wrapper list structure. In general, the filename is set for you by the [initWithURL:options:error:](#) (page 15), [initWithDirectoryWithFileWrappers:](#) (page 13), or [writeToURL:options:originalContentsURL:error:](#) (page 24) methods; you do not normally have to call this method directly.

Special Considerations

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `filename`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [filename](#) (page 11)
- [setPreferredFilename:](#) (page 23)
- [initWithURL:options:error:](#) (page 15)
- [initWithDirectoryWithFileWrappers:](#) (page 13)
- [writeToURL:options:originalContentsURL:error:](#) (page 24)

Declared In

NSFileWrapper.h

setIcon:

Specifies an image that can be used to represent the file wrapper to the user.

```
- (void)setIcon:(NSImage *)icon
```

Parameters*icon*

Image that can be used to represent the file wrapper to the user.

Discussion

An application does not have to use this icon; for example, a file viewer typically looks up icons automatically based on file extensions, and so wouldn't need this one. Similarly, if a file wrapper represents an image file, an application can display the image directly rather than a file icon.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [icon](#) (page 12)

Related Sample Code

CoreRecipes

Declared In

NSFileWrapper.h

setPreferredFilename:

Specifies the receiver's preferred filename.

```
- (void)setPreferredFilename:(NSString *)filename
```

Parameters*filename*

Preferred filename for the receiver.

Discussion

When a file wrapper is added to a parent directory file wrapper, the parent attempts to use the child's preferred filename as the key in its dictionary of children. If that key is already in use, then the parent derives a unique filename from the preferred filename and uses that for the key.

When you change the preferred filename of a file wrapper, the default implementation of this method causes existing parent directory file wrappers to remove and re-add the child to accommodate the change. Preferred filenames of children are not preserved when you write a file wrapper to disk and then later instantiate another file wrapper by reading the file from disk. If you need to preserve the user-visible names of attachments, you have to store the names yourself.

Special Considerations

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `filename`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [preferredFilename](#) (page 19)
- [setFilename:](#) (page 22)
- [addFileWrapper:](#) (page 8)
- [initWithURL:options:error:](#) (page 15)
- [initDirectoryWithFileWrappers:](#) (page 13)

- [writeToURL:options:originalContentsURL:error:](#) (page 24)

Related Sample Code

File Wrappers with Core Data Documents

GLUT

Declared In

NSFileWrapper.h

symbolicLinkDestinationURL

Provides the URL referenced by the receiver, which must be a symbolic-link file wrapper.

- (NSURL *)symbolicLinkDestinationURL

Return Value

Pathname the file wrapper references (that is, the destination of the symbolic link the file wrapper represents).

Discussion

This method may return `nil` if the user modifies the symbolic link after you call [readFromURL:options:error:](#) (page 19) or [initWithURL:options:error:](#) (page 15) but before `NSFileWrapper` has read the contents of the link. Use the `NSFileWrapperReadingImmediate` reading option to reduce the likelihood of that problem.

Special Considerations

This method raises `NSInternalInconsistencyException` if the receiver is not a symbolic-link file wrapper.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSFileWrapper.h

writeToURL:options:originalContentsURL:error:

Recursively writes the entire contents of a file wrapper to a given file-system URL.

- (BOOL)writeToFile:(NSURL *)url options:(NSFileWrapperWritingOptions)options
originalContentsURL:(NSURL *)originalContentsURL error:(NSError **)outError

Parameters

url

URL of the file-system node to which the file wrapper's contents are written.

options

Option flags for writing to the node located at *url*. See [“File Wrapper Writing Options”](#) (page 27) for possible values.

originalContentsURL

The location of a previous revision of the contents being written. The default implementation of this method attempts to avoid unnecessary I/O by writing hard links to regular files instead of actually writing out their contents when the contents have not changed. The child file wrappers must return accurate values when sent the `filename` (page 11) method for this to work. Use the `NSFileWrapperWritingWithNameUpdating` writing option to increase the likelihood of that.

Specify `nil` for this parameter if there is no earlier version of the contents or if you want to ensure that all the contents are written to files.

updateNames

YES to update the receiver's filenames (its filename and—for directory file wrappers—the filenames of its sub-file wrappers) be changed to the filenames of the corresponding nodes in the file system, after a successful write operation. Use this in Save or Save As operations.

NO to specify that the receiver's filenames not be updated. Use this in Save To operations.

outError

If an error occurs, upon return contains an `NSError` object that describes the problem. Pass `NULL` if you do not want error information.

Return Value

YES when the write operation is successful. If not successful, returns NO after setting `outError` to an `NSError` object that describes the reason why the file wrapper's contents could not be written.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [filename](#) (page 11)
- [readFromURL:options:error:](#) (page 19)

Declared In

`NSFileWrapper.h`

Constants

File Wrapper Reading Options

Reading options that can be set by the [initWithURL:options:error:](#) (page 15) and [readFromURL:options:error:](#) (page 19) methods.

```
enum {
    NSFileWrapperReadingImmediate = 1 << 0,
    NSFileWrapperReadingWithoutMapping = 1 << 1
};
typedef NSUInteger NSFileWrapperReadingOptions;
```

Constants

`NSFileWrapperReadingImmediate`

Whether the contents and icons are read immediately; applied recursively in the case of directory file wrappers.

If reading with this option succeeds, then subsequent invocations of `fileWrappers` (page 11), `regularFileContents` (page 20), `symbolicLinkDestinationURL` (page 24), `icon` (page 12), and `serializedRepresentation` (page 21) sent to the file wrapper and all its child file wrappers will fail and return `nil` only if an actual error occurs (for example, the volume has disappeared or the file server is unreachable)—not as a result of the user moving or deleting files.

For performance reasons, `NSFileWrapper` may not read the contents of some file packages immediately even when this option is chosen. For example, because the contents of bundles (not all file packages are bundles) are immutable to the user, `NSFileWrapper` may read the children of such a directory lazily.

You can use this option to take a snapshot of a file or folder for writing later. For example, an application like TextEdit can use this option when creating new file wrappers to represent attachments that the user creates by copying and pasting or dragging and dropping from the Finder to a TextEdit document. Don't use this option when reading a document file package, because that would cause unnecessarily bad performance. For example, an application wouldn't use this option when creating file wrappers to represent attachments as it's opening a document stored in a file package.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

`NSFileWrapperReadingWithoutMapping`

Whether file mapping for regular file wrappers is disallowed.

You can use this option to keep `NSFileWrapper` from memory-mapping files. This is useful if you want to make sure your application doesn't hold files open (mapped files are open files), therefore preventing the user from ejecting DVDs, unmounting disk partitions, or unmounting disk images. In Mac OS X v10.6 and later, `NSFileWrapper` memory-maps files that are on internal drives only. It never memory-maps files on external drives or network volumes, regardless of whether this option is used.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

Discussion

You can use the `NSFileWrapperReadingImmediate` and `NSFileWrapperReadingWithoutMapping` reading options together to take an exact snapshot of a file-system hierarchy that is safe from all errors (including the ones mentioned above) once reading has succeeded. If reading with both options succeeds, then subsequent invocations of the methods listed in the comment for the `NSFileWrapperReadingImmediate` reading option to the receiver and all its descendant file wrappers will never fail. However, note that reading with both options together is expensive in terms of both I/O and memory for large files, or directories containing large files, or even directories containing many small files.

File Wrapper Writing Options

Writing options that can be set by the [writeToURL:options:originalContentsURL:error:](#) (page 24) method.

```
enum {
    NSFileWrapperWritingAtomic = 1 << 0,
    NSFileWrapperWritingWithNameUpdating = 1 << 1
};
typedef NSUInteger NSFileWrapperWritingOptions;
```

Constants

`NSFileWrapperWritingAtomic`

Whether writing is done atomically.

You can use this option to ensure that, when overwriting a file package, the overwriting either completely succeeds or completely fails, with no possibility of leaving the file package in an inconsistent state. Because this option causes additional I/O, you shouldn't use it unnecessarily. For example, don't use this option in an override of `-[NSDocument writeToURL:ofType:error:]`, because `NSDocument safe-saving` is already done atomically.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

`NSFileWrapperWritingWithNameUpdating`

Whether descendant file wrappers are sent the [setFilename:](#) (page 22) method if the writing succeeds.

This option is necessary when your application passes a URL in the `originalContentsURL` parameter to the [writeToURL:options:originalContentsURL:error:](#) (page 24) method. Without using this option (and reusing child file wrappers properly), subsequent invocations of [writeToURL:options:originalContentsURL:error:](#) (page 24) would not be able to reliably create hard links in a new file package, because the record of names in the old file package would be out of date.

Available in Mac OS X v10.6 and later.

Declared in `NSFileWrapper.h`.

Deprecated NSFileWrapper Methods

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

Deprecated in Mac OS X v10.6

addFileWithPath:

Creates a file wrapper from a given file-system node and adds it to the receiver, which must be a directory file wrapper. (Deprecated in Mac OS X v10.6. Use [addFileWrapper:](#) (page 8) instead.)

```
- (NSString *)addFileWithPath:(NSString *)node
```

Parameters

node

file-system node from which to create the file wrapper to add to the directory.

Return Value

Dictionary key used to store the new file wrapper in the directory's list of file wrappers. See [Working With Directory Wrappers](#) for more information.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Instead of using this method, you can instantiate `NSFileWrapper` with one of the initializers, send it [setPreferredFilename:](#) (page 23) if necessary, and pass the result to [addFileWrapper:](#) (page 8).

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [addRegularFileWithContents:preferredFilename:](#) (page 9)
- [addSymbolicLinkWithDestination:preferredFilename:](#) (page 30)
- [removeFileWrapper:](#) (page 20)
- [fileWrappers](#) (page 11)

Related Sample Code

[File Wrappers with Core Data Documents](#)

Declared In

`NSFileWrapper.h`

addSymbolicLinkWithDestination:preferredFilename:

Creates a symbolic-link file wrapper pointing to a given file-system node and adds it to the receiver, which must be a directory file wrapper. (Deprecated in Mac OS X v10.6. Use [addFileWrapper:](#) (page 8) instead.)

```
- (NSString *)addSymbolicLinkWithDestination:(NSString *)node
    preferredFilename:(NSString *)preferredFilename
```

Parameters

node

Pathname the new symbolic-link file wrapper is to reference.

preferredFilename

Preferred filename for the new symbolic-link file wrapper.

Return Value

Dictionary key used to store the new file wrapper in the directory's list of file wrappers. See [Working With Directory Wrappers](#) for more information.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Instead of using this method, you can instantiate `NSFileWrapper` with one of the initializers, send it [setPreferredFilename:](#) (page 23) if necessary, and pass the result to [addFileWrapper:](#) (page 8).

This method raises `NSInternalInconsistencyException` if the receiver is not a directory file wrapper.

This method raises `NSInvalidArgumentException` if you pass `nil` or an empty value for `preferredFilename`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [addFileWrapper:](#) (page 8)
- [addRegularFileWithContents:preferredFilename:](#) (page 9)
- [removeFileWrapper:](#) (page 20)
- [fileWrappers](#) (page 11)

Declared In

`NSFileWrapper.h`

initWithSymbolicLinkWithDestination:

Initializes the receiver as a symbolic-link file wrapper. (Deprecated in Mac OS X v10.6. Use [initWithSymbolicLinkWithDestinationURL:](#) (page 14) instead.)

```
- (id)initWithSymbolicLinkWithDestination:(NSString *)node
```

Parameters

node

Pathname the receiver is to represent.

Return Value

Initialized symbolic-link file wrapper referencing *node*.

Discussion

The receiver is not associated to a file-system node until you save it using [writeToFile:atomically:updateFileNames:](#) (page 33). It's also initialized with open permissions; anyone can read or write the disk representations it saves.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of [initWithSymbolicLinkWithDestinationURL:](#) (page 14).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)

Declared In

NSFileWrapper.h

initWithPath:

Initializes a file wrapper instance whose kind is determined by the type of file-system node located by the path. (Deprecated in Mac OS X v10.6. Use [initWithURL:options:error:](#) (page 15) instead.)

```
- (id)initWithPath:(NSString *)node
```

Parameters

node

Pathname of the file-system node the file wrapper is to represent.

Return Value

File wrapper for *node*.

Discussion

If *node* is a directory, this method recursively creates file wrappers for each node within that directory.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of [initWithURL:options:error:](#) (page 15).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [setPreferredFilename:](#) (page 23)
- [filename](#) (page 11)
- [fileAttributes](#) (page 10)

Declared In

NSFileWrapper.h

needsToBeUpdatedFromPath:

Indicates whether the file wrapper needs to be updated to match a given file-system node. (Deprecated in Mac OS X v10.6. Use [matchesContentsOfURL:](#) (page 18) instead.)

- (BOOL)needsToBeUpdatedFromPath:(NSString *)node

Parameters

node

file-system node with which to compare the file wrapper.

Return Value

YES when the file wrapper needs to be updated to match *node*, NO otherwise.

Discussion

This table describes which attributes of the file wrapper and *node* are compared to determine whether the file wrapper needs to be updated:

File-wrapper type	Comparison determinants
Regular file	Modification date and access permissions.
Directory	Member hierarchy (recursive).
Symbolic link	Destination pathname.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of [matchesContentsOfURL:](#) (page 18).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [updateFromPath:](#) (page 33)
- [fileAttributes](#) (page 10)

Declared In

NSFileWrapper.h

symbolicLinkDestination

Provides the pathname referenced by the receiver, which must be a symbolic-link file wrapper. (Deprecated in Mac OS X v10.6. Use [symbolicLinkDestinationURL](#) (page 24) instead.)

- (NSString *)symbolicLinkDestination

Return Value

Pathname the file wrapper references (the destination of the symbolic link the file wrapper represents).

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of [symbolicLinkDestinationURL](#) (page 24).

This method raises `NSInternalInconsistencyException` if the receiver is not a symbolic-link file wrapper.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

Declared In

`NSFileWrapper.h`

updateFromPath:

Updates the file wrapper to match a given file-system node. (Deprecated in Mac OS X v10.6. Use [readFromURL:options:error:](#) (page 19) instead.)

- (BOOL)updateFromPath:(NSString *)*path*

Return Value

YES if the update is carried out, NO if it isn't needed.

Discussion

For a directory file wrapper, the contained file wrappers are also sent `updateFromPath:` messages. If nodes in the corresponding directory on the file system have been added or removed, corresponding file wrappers are released or created as needed.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of [readFromURL:options:error:](#) (page 19).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [needsToBeUpdatedFromPath:](#) (page 32)
- [updateAttachmentsFromPath:](#) (NSAttributedString)

Declared In

`NSFileWrapper.h`

writeToFile:atomically:updateFileNames:

Writes a file wrapper's contents to a given file-system node. (Deprecated in Mac OS X v10.6. Use [writeToURL:options:originalContentsURL:error:](#) (page 24) instead.)

- (BOOL)writeToFile:(NSString *)*node* atomically:(BOOL)*atomically*
updateFileNames:(BOOL)*updateNames*

Deprecated NSFileWrapper Methods

Parameters*node*

Pathname of the file-system node to which the receiver's contents are written.

atomically

YES to write the file safely so that:

- An existing file is not overwritten
- The method fails if the file cannot be written in its entirety

NO to overwrite an existing file and ignore incomplete writes.

updateNames

YES to update the receiver's filenames (its filename and—for directory file wrappers—the filenames of its sub-file wrappers) be changed to the filenames of the corresponding nodes in the file system, after a successful write operation. Use this in Save or Save As operations.

NO to specify that the receiver's filenames not be updated. Use this in Save To operations.

Return Value

YES when the write operation is successful, NO otherwise.

Special Considerations

Beginning with Mac OS X v10.6, the preferred method of referring to files is with a `file://` URL. Therefore, this method has been deprecated in favor of `writeToURL:options:originalContentsURL:error:` (page 24).

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- [filename](#) (page 11)
- [writeToURL:options:originalContentsURL:error:](#) (page 24)

Related Sample Code

File Wrappers with Core Data Documents
Quartz Composer WWDC 2005 TextEdit

Declared In

NSFileWrapper.h

Document Revision History

This table describes the changes to *NSFileWrapper Class Reference*.

Date	Notes
2009-05-26	Updated for Mac OS X v10.6. Added methods that use file system URLs instead of file paths.
	Methods using pathnames were deprecated and replaced with methods using URLs.
2008-10-15	Corrected -initWithSerializedRepresentation: method description.
2007-03-27	Made editorial improvements.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`addFilePath:` [instance method 29](#)
`addFileWrapper:` [instance method 8](#)
`addRegularFileWithContents:preferredFilename:`
[instance method 9](#)
`addSymbolicLinkWithDestination:preferredFilename:`
[instance method 30](#)

F

`File Wrapper Reading Options` [25](#)
`File Wrapper Writing Options` [27](#)
`fileAttributes` [instance method 10](#)
`filename` [instance method 11](#)
`fileWrappers` [instance method 11](#)

I

`icon` [instance method 12](#)
`initWithDirectoryWithFileWrappers:` [instance method 13](#)
`initWithRegularFileWithContents:` [instance method 13](#)
`initWithSymbolicLinkWithDestination:` [instance method 30](#)
`initWithSymbolicLinkWithDestinationURL:` [instance method 14](#)
`initWithPath:` [instance method 31](#)
`initWithSerializedRepresentation:` [instance method 15](#)
`initWithURL:options:error:` [instance method 15](#)
`isDirectory` [instance method 16](#)
`isRegularFile` [instance method 16](#)
`isSymbolicLink` [instance method 17](#)

K

`keyForFileWrapper:` [instance method 17](#)

M

`matchesContentsOfURL:` [instance method 18](#)

N

`needsToBeUpdatedFromPath:` [instance method 32](#)
`NSFileWrapperReadingImmediate` [constant 26](#)
`NSFileWrapperReadingWithoutMapping` [constant 26](#)
`NSFileWrapperWritingAtomic` [constant 27](#)
`NSFileWrapperWritingWithNameUpdating` [constant 27](#)

P

`preferredFilename` [instance method 19](#)

R

`readFromURL:options:error:` [instance method 19](#)
`regularFileContents` [instance method 20](#)
`removeFileWrapper:` [instance method 20](#)

S

`serializedRepresentation` [instance method 21](#)
`setFileAttributes:` [instance method 21](#)
`setFilename:` [instance method 22](#)
`setIcon:` [instance method 22](#)
`setPreferredFilename:` [instance method 23](#)

`symbolicLinkDestination` **instance method** [32](#)
`symbolicLinkDestinationURL` **instance method** [24](#)

U

`updateFromPath:` **instance method** [33](#)

W

`writeToFile:atomically:updateFileNames:`
instance method [33](#)
`writeToURL:options:originalContentsURL:error:`
instance method [24](#)