NSWorkspace Class Reference

Cocoa > Interapplication Communication

2007-04-27

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

Simultaneously published in the United States and Canada.

Contents

NSWorkspace Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Workspace Services Programming Topics
Declared in	NSWorkspace.h
Related sample code	CocoaDVDPlayer MyPhoto ObjectPath Quartz Composer WWDC 2005 TextEdit TextEditPlus

Overview

An `NSWorkspace` object responds to application requests to perform a variety of services:

- Opening, manipulating, and obtaining information about files and devices
- Tracking changes to the file system, devices, and the user database
- Launching applications

There is one shared `NSWorkspace` object per application. You use the class method `sharedWorkspace` (page 11) to access it. For example, the following statement uses an `NSWorkspace` object to request that a file be opened in the TextEdit application:

```
[[NSWorkspace sharedWorkspace] openFile:@"/Myfiles/README"
    withApplication:@"TextEdit"];
```

Tasks

Accessing the Shared NSWorkspace Instance

`+ sharedWorkspace` (page 11)
Returns the shared `NSWorkspace` instance.

Accessing the NSWorkspace Notification Center

`- notificationCenter` (page 24)
Returns the notification center for workspace notifications.

Opening Files

`- openFile:` (page 24)
Opens the specified file specified using the default application associated with its type.

`- openFile:withApplication:` (page 25)
Opens a file using the specified application.

`- openFile:fromImage:at:inView:` (page 25)
Opens a file using the default application for its type and animates the action using a custom icon.

`- openFile:withApplication:andDeactivate:` (page 26)
Opens the specified file and optionally deactivates the sending application.

`- openTempFile:` (page 27)
Opens the specified temporary file using the default application for its type.

`- openURL:` (page 27)
Opens the location at the specified URL.

Manipulating Applications

`- launchApplication:` (page 18)
Launches the specified application.

`- launchApplication:showIcon:autolaunch:` (page 19)
Launches the specified application using additional options.

`- hideOtherApplications` (page 16)
Hides all applications other than the sender.

Manipulating Files

`- performFileOperation:source:destination:files:tag:` (page 29)
Performs a file operation on a set of files in a particular directory.

- `selectFile:inFileViewerRootedAtPath:` (page 30)
  Selects the file specified by *fullPath*.

Manipulating Uniform Type Identifier Information

- `typeOfFile:error:` (page 32)
  Returns the uniform type identifier of the specified file, if it can be determined..
- `localizedDescriptionForType:` (page 20)
  Returns the localized description for the specified Uniform Type Identifier
- `preferredFilenameExtensionForType:` (page 30)
  Returns the preferred filename extension for the specified Uniform Type Identifier.
- `fileNameExtension:isValidForType:` (page 13)
  Returns whether the specified filename extension is appropriate for the Uniform Type Identifier.
- `type:conformsToType:` (page 32)
  Returns a Boolean indicating that the first Uniform Type Identifier conforms to the second Uniform Type Identifier.

Requesting Information

- `iconForFile:` (page 16)
  Returns an image containing the icon for the specified file.
- `iconForFileType:` (page 17)
  Returns an image containing the icon for files of the specified type.
- `iconForFiles:` (page 17)
  Returns an image containing the icon for the specified files.
- `getInfoForFile:application:type:` (page 15)
  Retrieves information about the specified file.
- `fullPathForApplication:` (page 14)
  Returns the full path for the specified application.
- `getFileSystemInfoForPath:isRemovable:isWritable:isUnmountable:description:type:` (page 14)
  Describes the file system at *fullPath*.
- `isFilePackageAtPath:` (page 18)
  Determines whether the specified path is a file package.
- `activeApplication` (page 12)
  Returns a dictionary with information about the current active application.
- `launchedApplications` (page 20)
  Returns an array of dictionaries, one entry for each running application.

Image Animation

- `slideImage:from:to:` (page 31)
  Animates a sliding image from one point to another.

Requesting Additional Time Before Logout

- `extendPowerOffBy:` (page 13)
  Requests the system wait for the specified amount of time before turning off the power or logging out the user.

Tracking Changes to the File System

- `noteFileSystemChanged` (page 22)
  Informs the `NSWorkspace` object that the file system has changed.
- `noteFileSystemChanged:` (page 23)
  Informs the `NSWorkspace` object that the file system changed at the specified path.
- `fileSystemChanged` (page 14)
  Returns a Boolean value indicating whether a change to the file system has been registered with a `noteFileSystemChanged` (page 22) message since the last `fileSystemChanged` (page 14) message.

Updating Registered Services and File Types

- `findApplications` (page 14)
  Examines all applications and updates the records of registered services and file types.

Tracking Changes to the Defaults Database

- `noteUserDefaultsChanged` (page 23)
  Informs the `NSWorkspace` object that the defaults database has changed.
- `userDefaultsChanged` (page 33)
  Returns a Boolean value indicating whether a change to the defaults database has been registered with a `noteUserDefaultsChanged` (page 23) message since the last `userDefaultsChanged` (page 33) message.

Tracking Status Changes for Applications and Devices

- `mountedRemovableMedia` (page 21)
  Returns the full pathnames of all currently mounted removable disks.
- `mountNewRemovableMedia` (page 22)
  Returns the full pathnames of any newly mounted disks.
- `mountedLocalVolumePaths` (page 21)
  Returns the mount points of all local volumes, not just the removable ones returned by `mountedRemovableMedia` (page 21).
- `checkForRemovableMedia` (page 12)
  Polls the system's drives for any disks that have been inserted but not yet mounted.

Providing Custom Icons

`- setIcon:forFile:options:` (page 31)
Sets the icon for the file or directory at the specified path.

Unmounting a Device

`- unmountAndEjectDeviceAtPath:` (page 33)
Unmounts and ejects the device at the specified path.

Working with Bundles

`- absolutePathForAppBundleWithIdentifier:` (page 12)
Returns the absolute file-system path of an application bundle.

`- launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:` (page 19)
Launches the application corresponding to the specified *bundleIdentifier*.

`- openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:` (page 28)
Opens one or more files from an array of URLs.

Class Methods

sharedWorkspace

Returns the shared `NSWorkspace` instance.

```
+ (NSWorkspace *)sharedWorkspace
```

Return Value
The `NSWorkspace` object associated with the process.

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

Related Sample Code
CocoaDVDPlayer
MyPhoto
ObjectPath
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSWorkspace.h`

Instance Methods

absolutePathForAppBundleWithIdentifier:

Returns the absolute file-system path of an application bundle.

```
- (NSString *)absolutePathForAppBundleWithIdentifier:(NSString *)bundleIdentifier
```

Parameters

bundleIdentifier

The bundle identifier string. This value corresponds to the value in the `CFBundleIdentifier` key of the application's `Info.plist` file. For example, the bundle identifier of the TextEdit application is `com.apple.TextEdit`.

Return Value

The file system path to the application bundle identified by *bundleIdentifier*, or `nil` if the bundle cannot be found.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSWorkspace.h`

activeApplication

Returns a dictionary with information about the current active application.

```
- (NSDictionary *)activeApplication
```

Return Value

A dictionary with information about the application. The dictionary contains as many of the keys described in "Constants" (page 34) as are available.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- launchedApplications` (page 20)

Declared In

`NSWorkspace.h`

checkForRemovableMedia

Polls the system's drives for any disks that have been inserted but not yet mounted.

```
- (void)checkForRemovableMedia
```

Discussion
This method doesn't wait until such disks are mounted; instead, it requests that the disk be mounted asynchronously and returns immediately. Currently has no effect.

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

See Also
- `mountNewRemovableMedia` (page 22)
- `mountedRemovableMedia` (page 21)

Declared In
`NSWorkspace.h`

extendPowerOffBy:

Requests the system wait for the specified amount of time before turning off the power or logging out the user.

```
- (NSInteger)extendPowerOffBy:(NSInteger)requested
```

Parameters
requested
The number of milliseconds to wait before turning off the power or logging off the user.

Return Value
The number of milliseconds granted by the system.

Discussion
Currently unimplemented.

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

Declared In
`NSWorkspace.h`

fileNameExtension:isValidForType:

Returns whether the specified filename extension is appropriate for the Uniform Type Identifier.

```
- (BOOL)filenameExtension:(NSString *)filenameExtension isValidForType:(NSString
    *)typeName
```

Parameters
filenameExtension
A string containing the filename extension.

typeName
A string containing the Uniform Type Identifier.

Return Value
`YES` if *fileNameExtension* is a valid extension for *typeName*, `NO` otherwise

fileSystemChanged

Returns a Boolean value indicating whether a change to the file system has been registered with a `noteFileSystemChanged` (page 22) message since the last `fileSystemChanged` (page 14) message.

```
- (BOOL)fileSystemChanged
```

Return Value
Currently, this method always returns `NO`.

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

Declared In
`NSWorkspace.h`

findApplications

Examines all applications and updates the records of registered services and file types.

```
- (void)findApplications
```

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

Declared In
`NSWorkspace.h`

fullPathForApplication:

Returns the full path for the specified application.

```
- (NSString *)fullPathForApplication:(NSString *)appName
```

Parameters

appName
The name of the application.

Return Value
The full path for the application, or `nil` if the specified application was not found.

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

Declared In
`NSWorkspace.h`

getFileSystemInfoForPath:isRemovable:isWritable:isUnmountable:description:type:

Describes the file system at *fullPath*.

```
- (BOOL)getFileSystemInfoForPath:(NSString *)fullPath isRemovable:(BOOL
    *)removableFlag isWritable:(BOOL *)writableFlag isUnmountable:(BOOL
    *)unmountableFlag description:(NSString **)description type:(NSString
    **)fileSystemType
```

Parameters

fullPath

The path to the file-system mount point.

removableFlag

On input, a boolean variable; on return, this variable contains YES if the file system is on removable media.

writableFlag

On input, a boolean variable; on return, this variable contains YES if the file system writable.

unmountableFlag

On input, a boolean variable; on return, this variable contains YES if the file system is unmountable.

description

On input, a pointer to a string object variable; on return, if the method was successful, this variable contains a string object that describes the file system. You should not rely on this description for program logic but can use it in message strings. Values can include “hard,” “nfs,” and “foreign."

fileSystemType

On input, a pointer to a string object variable; on return, if the method was successful, this variable contains the file-system type. Values can include “HFS,” “UFS,” or other values.

Return Value

YES if the information was successfully returned, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSWorkspace.h

getInfoForFile:application:type:

Retrieves information about the specified file.

```
- (BOOL)getInfoForFile:(NSString *)fullPath application:(NSString **)appName
    type:(NSString **)type
```

Parameters

fullPath

The full path to the desired file.

appName

The application the system would use to open the file.

type

On input, a pointer to a string object variable; on return, if the method is successful, this variable contains a string object with the filename extension or encoded HFS file type of the file.

Return Value

YES if the information was retrieved successfully; otherwise, NO if the file could not be found or the application was not associated with the file.

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

See Also
- `iconForFile:` (page 16)
- `iconForFiles:` (page 17)

Declared In
`NSWorkspace.h`

hideOtherApplications

Hides all applications other than the sender.

```
- (void)hideOtherApplications
```

Discussion
The user can hide all applications except the current one by Command-Option-clicking on an application's Dock icon.

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

Declared In
`NSWorkspace.h`

iconForFile:

Returns an image containing the icon for the specified file.

```
- (NSImage *)iconForFile:(NSString *)fullPath
```

Parameters
fullPath
The full path to the file.

Return Value
The icon associated with the file.

Discussion
The returned image has an initial size of 32 pixels by 32 pixels.

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

See Also
- `getInfoForFile:application:type:` (page 15)
- `iconForFileType:` (page 17)
- `iconForFiles:` (page 17)

Related Sample Code
bMoviePalette
bMoviePaletteCocoa

SpotlightAPI

Declared In
NSWorkspace.h

iconForFiles:

Returns an image containing the icon for the specified files.

```
- (NSImage *)iconForFiles:(NSArray *)fullPaths
```

Parameters

fullPaths

An array of NSString objects, each of which contains the full path to a file.

Return Value
The icon associated with the group of files.

Discussion
If *fullPaths* specifies one file, that file's icon is returned. If *fullPaths* specifies more than one file, an icon representing the multiple selection is returned.

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

See Also
- iconForFile: (page 16)
- iconForFileType: (page 17)

Declared In
NSWorkspace.h

iconForFileType:

Returns an image containing the icon for files of the specified type.

```
- (NSImage *)iconForFileType:(NSString *)fileType
```

Parameters

fileType

The file type, which may be either a filename extension or an encoded HFS file type.

Return Value
The icon associated with files of the given type.

Discussion
The returned image has an initial size of 32 pixels by 32 pixels.

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

See Also
- iconForFile: (page 16)
- iconForFiles: (page 17)

Related Sample Code
MyPhoto
ObjectPath

Declared In
NSWorkspace.h

isFilePackageAtPath:

Determines whether the specified path is a file package.

```
- (BOOL)isFilePackageAtPath:(NSString *)fullPath
```

Parameters
fullPath
The full path to examine.

Return Value
YES if the path identifies a file package; otherwise, NO if the path does not exist, is not a directory, or is not a file package.

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

Declared In
NSWorkspace.h

launchApplication:

Launches the specified application.

```
- (BOOL)launchApplication:(NSString *)appName
```

Parameters
appName
The name of the application to open.

Return Value
YES if the application was successfully launched or was already running; otherwise, NO.

Discussion
The *appName* parameter need not be specified with a full path and, in the case of an application wrapper, may be specified with or without the .app extension, as described in "Use of .app Extension".

Before this method begins, it posts an NSWorkspaceWillLaunchApplicationNotification (page 41) to the NSWorkspace object's notification center. When the operation is complete, it posts an NSWorkspaceDidLaunchApplicationNotification (page 39).

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

See Also
- launchApplication:showIcon:autolaunch: (page 19)

Related Sample Code
QTAudioExtractionPanel

Declared In
NSWorkspace.h

launchApplication:showIcon:autolaunch:

Launches the specified application using additional options.

```
- (BOOL)launchApplication:(NSString *)appName showIcon:(BOOL)showIcon
    autolaunch:(BOOL)autolaunch
```

Parameters

appName
The name of the application to open.

showIcon
If NO, the application's icon is not placed on the screen. (The icon still exists, though.)

autolaunch
If YES, the autolaunch default is set as though the specified application were autolaunched at startup.

Return Value
YES if the application was successfully launched or was already running; otherwise, NO.

Discussion
This method is provided to enable daemon-like applications that lack a normal user interface. Its use is not generally encouraged.

Returns YES if the application is successfully launched or already running, and NO if it can't be launched.

Before this method begins, it posts an NSWorkspaceWillLaunchApplicationNotification (page 41) to the NSWorkspace object's notification center. When the operation is complete, it posts an NSWorkspaceDidLaunchApplicationNotification (page 39).

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

See Also
- launchApplication: (page 18)

Declared In
NSWorkspace.h

launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:

Launches the application corresponding to the specified *bundleIdentifier*.

```
- (BOOL)launchAppWithBundleIdentifier:(NSString *)bundleIdentifier
    options:(NSWorkspaceLaunchOptions)options
    additionalEventParamDescriptor:(NSAppleEventDescriptor *)descriptor
    launchIdentifier:(NSNumber **)identifier
```

Parameters

bundleIdentifier

A bundle identifier string. This value corresponds to the value in the `CFBundleIdentifier` key of the application's `Info.plist` file. For example, the bundle identifier of the TextEdit application is `com.apple.TextEdit`.

options

Options to use when launching the application. Values for this parameter are described in "Constants" (page 34).

descriptor

Additional options specified in an AppleEvent-style descriptor. For example, you could use this parameter to specify additional documents to open when the application is launched.

identifier

On input, a pointer to a number object variable. On return, the variable contains a number object with a unique identifier for the launch attempt. You can use this value to distinguish individual launch requests. This parameter may be `nil`.

Return Value

`YES` if the application was found and launched; otherwise, `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:` (page 28)

Declared In

`NSWorkspace.h`

launchedApplications

Returns an array of dictionaries, one entry for each running application.

```
- (NSArray *)launchedApplications
```

Return Value

An array of `NSDictionary` objects. Each dictionary contains as many of the keys described in "Constants" (page 34) as are available.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- activeApplication` (page 12)

Declared In

`NSWorkspace.h`

localizedDescriptionForType:

Returns the localized description for the specified Uniform Type Identifier

```
- (NSString *)localizedDescriptionForType:(NSString *)typeName
```

Parameters

typeName

A string containing the Uniform Type Identifier.

Return Value

An NSString containg the localized description of *typeName*.

Discussion

The localized description is suitable for displaying to the user.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSWorkspace.h

mountedLocalVolumePaths

Returns the mount points of all local volumes, not just the removable ones returned by `mountedRemovableMedia` (page 21).

```
- (NSArray *)mountedLocalVolumePaths
```

Return Value

An array of `NSString` objects, each of which contains the full pathname of the mount point for any local volumes.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSWorkspace.h

mountedRemovableMedia

Returns the full pathnames of all currently mounted removable disks.

```
- (NSArray *)mountedRemovableMedia
```

Return Value

An array of `NSString` objects, each of which contains the full pathname of a mounted removable disk.

Discussion

If the computer provides an interrupt or other notification when the user inserts a disk into a drive, the Finder will mount the disk immediately. However, if no notification is given, the Finder won't be aware that a disk needs to be mounted. On such systems, an application should invoke either `mountNewRemovableMedia` (page 22) or `checkForRemovableMedia` (page 12) before invoking `mountedRemovableMedia` (page 21). Either of these methods cause the Finder to poll the drives to see if a disk is present. If a disk has been inserted but not yet mounted, these methods will cause the Finder to mount it.

The Disk button in an Open or Save panel invokes `mountedRemovableMedia` (page 21) and `mountNewRemovableMedia` (page 22) as part of its operation, so most applications won't need to invoke these methods directly.

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

See Also
- `checkForRemovableMedia` (page 12)
- `mountNewRemovableMedia` (page 22)

Related Sample Code
CocoaDVDPlayer

Declared In
`NSWorkspace.h`

mountNewRemovableMedia

Returns the full pathnames of any newly mounted disks.

```
- (NSArray *)mountNewRemovableMedia
```

Return Value
An array of NSString objects, each of which contains the full pathname to a newly mounted disk.

Discussion
This method polls the system's drives for any disks that have been inserted but not yet mounted and waits until the new disks have been mounted. This method posts an `NSWorkspaceDidMountNotification` (page 39) to the `NSWorkspace` object's notification center when it is finished. Currently provides the same functionality as `mountedRemovableMedia` (page 21).

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

See Also
- `checkForRemovableMedia` (page 12)
- `mountedRemovableMedia` (page 21)

Declared In
`NSWorkspace.h`

noteFileSystemChanged

Informs the `NSWorkspace` object that the file system has changed.

```
- (void)noteFileSystemChanged
```

Discussion
The `NSWorkspace` object then gets the status of all the files and directories it is interested in and updates itself appropriately. This method is used by many objects that write or delete files.

The NSDocument and NSSavePanel objects use this method when saving a file. If you create a file directly, you should call noteFileSystemChanged (page 22) so that the Finder can update the folder if it is open.

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

See Also
- fileSystemChanged (page 14)

Declared In
NSWorkspace.h

noteFileSystemChanged:

Informs the NSWorkspace object that the file system changed at the specified path.

```
- (void)noteFileSystemChanged:(NSString *)path
```

Parameters

path
The full path that changed.

Discussion
The NSWorkspace object then gets the status of all the files and directories it is interested in and updates itself appropriately. This method is used by many objects that write or delete files.

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

See Also
- fileSystemChanged (page 14)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
NSWorkspace.h

noteUserDefaultsChanged

Informs the NSWorkspace object that the defaults database has changed.

```
- (void)noteUserDefaultsChanged
```

Discussion
The NSWorkspace object then reads all the defaults it is interested in and reconfigures itself appropriately. For example, this method is used by the Preferences application to notify the Finder whether the user prefers to see hidden files. Currently has no effect.

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

See Also
- `userDefaultsChanged` (page 33)

Declared In
`NSWorkspace.h`

notificationCenter

Returns the notification center for workspace notifications.

```
- (NSNotificationCenter *)notificationCenter
```

Return Value
The notification center object.

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

Related Sample Code
CocoaDVDPlayer

Declared In
`NSWorkspace.h`

openFile:

Opens the specified file specified using the default application associated with its type.

```
- (BOOL)openFile:(NSString *)fullPath
```

Parameters
fullPath
The full path to the file.

Return Value
`YES` if the file was successfully opened; otherwise, `NO`.

Discussion
The sending application is deactivated before the request is sent.

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

See Also
- `openFile:fromImage:at:inView:` (page 25)
- `openFile:withApplication:` (page 25)
- `openFile:withApplication:andDeactivate:` (page 26)
- `openTempFile:` (page 27)

Related Sample Code
iSpend
QTRecorder
Quartz Composer WWDC 2005 TextEdit

TextEditPlus
UIElementInspector

Declared In
NSWorkspace.h

openFile:fromImage:at:inView:

Opens a file using the default application for its type and animates the action using a custom icon.

```
- (BOOL)openFile:(NSString *)fullPath fromImage:(NSImage *)anImage at:(NSPoint)point
    inView:(NSView *)aView
```

Parameters

fullPath
The full path to the file.

anImage
The icon for the file.

point
The point in `aView` at which to display the icon.

aView
The view in which to display the icon.

Return Value
`YES` if the file was successfully opened; otherwise, `NO`.

Discussion
The Finder provides an animation before opening the file to give the user feedback that the file is to be opened. To provide this animation, *anImage* should contain an icon for the file, and its image should be displayed at *point*, specified in the coordinates of *aView*. Currently provides the same functionality as `openFile:` (page 24).

The sending application is deactivated before the request is sent.

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

See Also
- `openFile:` (page 24)
- `openFile:withApplication:` (page 25)
- `openFile:withApplication:andDeactivate:` (page 26)
- `openTempFile:` (page 27)

Declared In
NSWorkspace.h

openFile:withApplication:

Opens a file using the specified application.

```
- (BOOL)openFile:(NSString *)fullPath withApplication:(NSString *)appName
```

Parameters

fullPath

The full path to the file.

appName

The name of the application to use when opening the file.

Return Value

YES if the file was successfully opened; otherwise, NO.

Discussion

The *appName* parameter need not be specified with a full path and, in the case of an application wrapper, may be specified with or without the .app extension, as described in "Use of .app Extension". The sending application is deactivated before the request is sent.

Availability

Available in Mac OS X v10.0 and later.

See Also

- openFile: (page 24)
- openFile:withApplication:andDeactivate: (page 26)

Related Sample Code

CIVideoDemoGL

ExtractMovieAudioToAIFF

QTExtractAndConvertToAIFF

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

NSWorkspace.h

openFile:withApplication:andDeactivate:

Opens the specified file and optionally deactivates the sending application.

```
- (BOOL)openFile:(NSString *)fullPath withApplication:(NSString *)appName
    andDeactivate:(BOOL)flag
```

Parameters

fullPath

The full path to the file.

appName

The name of the application to use when opening the file.

flag

If YES, the sending application is deactivated before the request is sent, allowing the opening application to become the active application.

Return Value

YES if the file was successfully opened; otherwise, NO.

Discussion
The *appName* parameter need not be specified with a full path and, in the case of an application wrapper, may be specified with or without the `.app` extension, as described in "Use of .app Extension". If *appName* is `nil`, the default application for the file's type is used.

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

See Also
- `openFile:` (page 24)
- `openFile:withApplication:` (page 25)
- `application:openFile:` (NSApplication delegate method)

Related Sample Code
Core Data HTML Store

Declared In
`NSWorkspace.h`

openTempFile:

Opens the specified temporary file using the default application for its type.

```
- (BOOL)openTempFile:(NSString *)fullPath
```

Parameters

fullPath
The full path to the temporary file.

Return Value
`YES` if the file was successfully opened; otherwise, `NO`.

Discussion
The sending application is deactivated before the request is sent. Using this method instead of one of the `openFile:...` methods lets the receiving application know that it should delete the file when it no longer needs it. Currently provides the same functionality as `openFile:` (page 24).

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

See Also
- `openFile:` (page 24)
- `openFile:fromImage:at:inView:` (page 25)
- `openFile:withApplication:` (page 25)
- `openFile:withApplication:andDeactivate:` (page 26)

Declared In
`NSWorkspace.h`

openURL:

Opens the location at the specified URL.

```
- (BOOL)openURL:(NSURL *)url
```

Parameters

url

A URL specifying the location to open.

Return Value

`YES` if the location was successfully opened; otherwise, `NO`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

NSOperationSample

ObjectPath

PredicateEditorSample

QT Capture Widget

VertexPerformanceTest

Declared In

`NSWorkspace.h`

openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:

Opens one or more files from an array of URLs.

```
- (BOOL)openURLs:(NSArray *)urls withAppBundleIdentifier:(NSString *)bundleIdentifier
    options:(NSWorkspaceLaunchOptions)options
   additionalEventParamDescriptor:(NSAppleEventDescriptor *)descriptor
   launchIdentifiers:(NSArray **)identifiers
```

Parameters

urls

An array of `NSURL` objects, each one identifying a URL for the application to open.

bundleIdentifier

A bundle identifier string or `nil` to use the default system bindings. This value corresponds to the value in the `CFBundleIdentifier` key of the application's `Info.plist` file. For example, the bundle identifier of the TextEdit application is `com.apple.TextEdit`.

options

Options to use when launching the application. Values for this parameter are described in "Constants" (page 34).

descriptor

Additional options specified in an AppleEvent-style descriptor. For example, you could use this parameter to specify additional documents to open when the application is launched.

identifiers

On input, a pointer to an array object variable. On return, the variable contains an array of NSNumber objects. Each number object contains a unique identifier (one for each URL) for the launch attempt. You can use these values to distinguish individual launch requests. This parameter may be `nil`.

Return Value

`YES` if the application was found and launched; otherwise, `NO`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:` (page 19)

Related Sample Code
NewsReader

Declared In
`NSWorkspace.h`

performFileOperation:source:destination:files:tag:

Performs a file operation on a set of files in a particular directory.

```
- (BOOL)performFileOperation:(NSString *)operation source:(NSString *)source
    destination:(NSString *)destination files:(NSArray *)files tag:(NSInteger *)tag
```

Parameters

operation
The file operation to perform. The possible values for this parameter are described in "Constants" (page 34).

source
The full path to the directory containing the files on which to operate.

destination
The full path to the destination directory of the operation.

files
An array of `NSString` objects specifying the names of the files and directories to be manipulated. Each string must not contain any path information other than the name of the file or directory. In other words, all of the files and directories must be located in the source directory and not in one if its subdirectories.

tag
On input, a integer variable; on return, this variable contains a negative integer if the operation fails, 0 if the operation was performed synchronously and succeeded, or a positive integer if the operation was performed asynchronously. If the value is a positive integer, the value is a tag that identifies the requested file operation.

Return Value
`YES` if the operation succeeded; otherwise, `NO`.

Discussion
Some operations—such as moving, copying, and linking files—require a destination directory to be specified. If not, *destination* should be the empty string (`@""`). Before this method returns, it posts an `NSWorkspaceDidPerformFileOperationNotification` (page 39) to the `NSWorkspace` object's notification center.

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

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSWorkspace.h`

preferredFilenameExtensionForType:

Returns the preferred filename extension for the specified Uniform Type Identifier.

```
- (NSString *)preferredFilenameExtensionForType:(NSString *)typeName
```

Parameters

typeName
A string containing the Uniform Type Identifier.

Return Value
The appropriate filename extension for typeName, or `nil` if no extension could be determined.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSWorkspace.h`

selectFile:inFileViewerRootedAtPath:

Selects the file specified by *fullPath*.

```
- (BOOL)selectFile:(NSString *)fullPath inFileViewerRootedAtPath:(NSString
    *)rootFullPath
```

Parameters

fullPath
The full path of the file to select.

rootFullPath
If a path is specified, a new file viewer is opened. If you specify an empty string (@"") for this parameter, the file is selected in the main viewer.

Return Value
`YES` if the file was successfully selected; otherwise, `NO`.

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

Related Sample Code
EnhancedAudioBurn

Declared In
`NSWorkspace.h`

setIcon:forFile:options:

Sets the icon for the file or directory at the specified path.

```
- (BOOL)setIcon:(NSImage *)image forFile:(NSString *)fullPath
    options:(NSWorkspaceIconCreationOptions)options
```

Parameters

image

The image to use as the icon for the file or directory.

fullPath

The full path of the file or directory.

options

The icon representations to generate from the image. You specify this value by combining the appropriate `NSWorkspaceIconCreationOptions` constants, listed in "Constants" (page 34), using the C bitwise `OR` operator. Specify `0` if you want to generate icons in all available icon representation formats.

Return Value

`YES` if the icon was set; otherwise, `NO`.

Discussion

The *image* can be an arbitrary image, with or without transparency. This image is automatically scaled (as needed) to generate the icon representations. The file or folder must exist and be writable by the user.

It is recommended that applications include the `NSExclude10_4ElementsIconCreationOption` option for compatibility with pre-Mac OS X v10.3 Finder. Icons that include the high resolution elements prevent custom icons from being displayed on earlier systems.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSWorkspace.h`

slideImage:from:to:

Animates a sliding image from one point to another.

```
- (void)slideImage:(NSImage *)image from:(NSPoint)fromPoint to:(NSPoint)toPoint
```

Parameters

image

The image to animate.

fromPoint

The starting point, in screen coordinates.

toPoint

The ending point, in screen coordinates.

Discussion

Currently unimplemented.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSWorkspace.h

type:conformsToType:

Returns a Boolean indicating that the first Uniform Type Identifier conforms to the second Uniform Type Identifier.

```
- (BOOL)type:(NSString *)firstTypeName conformsToType:(NSString *)secondTypeName
```

Parameters

firstTypeName
A string containing the Uniform Type Identifier that should conform to *secondTypeName*.

secondTypeName
A string containing a Uniform Type Identifier.

Return Value
YES if *firstTypeName* conforms to the uniform type identifier hierarchy of *secondTypeName*, NO otherwise.

Discussion
Use this method instead of comparing Uniform Identifier Types for equality. See *Uniform Type Identifiers Overview* for information about Uniform Type Identifier conformance.

This method will always return YES if the two strings are equal. It is appropriate to use this method with other type names, including those declared in CFBundleTypeName Info.plist entries.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSWorkspace.h

typeOfFile:error:

Returns the uniform type identifier of the specified file, if it can be determined..

```
- (NSString *)typeOfFile:(NSString *)absoluteFilePath error:(NSError **)outError
```

Parameters

absoluteFilePath
The absolute path of the file.

outError
If the Uniform Type Identifier of the file at absolutePath can't be determined, *outError* contains an NSError object that describing why.

Return Value
An NSString containing the uniform type identifier of the file at absoluteFilePath. If no UTI can be determined the return value is nil.

Discussion
If the file at the specified path is a symbolic link, the type of the symbolic link is returned.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSWorkspace.h

unmountAndEjectDeviceAtPath:

Unmounts and ejects the device at the specified path.

```
- (BOOL)unmountAndEjectDeviceAtPath:(NSString *)path
```

Parameters

path

The path to the device.

Return Value
YES if the device was unmounted; otherwise, NO.

Discussion
When this method begins, it posts an NSWorkspaceWillUnmountNotification (page 42) to the NSWorkspace object's notification center. When it is finished, it posts an NSWorkspaceDidUnmountNotification (page 40).

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

Related Sample Code
CocoaDVDPlayer

Declared In
NSWorkspace.h

userDefaultsChanged

Returns a Boolean value indicating whether a change to the defaults database has been registered with a noteUserDefaultsChanged (page 23) message since the last userDefaultsChanged (page 33) message.

```
- (BOOL)userDefaultsChanged
```

Return Value
Currently, this method always returns NO.

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

Declared In
NSWorkspace.h

Constants

The following table describes keys for an `NSDictionary` object containing information about an application. This dictionary is returned by `activeApplication` (page 12) and `launchedApplications` (page 20), and is also provided in the *`userInfo`* of `NSWorkspace` notifications for application launch and termination.

Key	Value
`@"NSApplicationPath"`	The full path to the application, as a `NSString` object.
`@"NSApplicationName"`	The application's name, as an `NSString` object.
`@"NSApplicationBundleIdentifier"`	The application's bundle identifier, as an `NSString` object.
`@"NSApplicationProcessIdentifier"`	The application's process id, as an `NSNumber` object.
`@"NSApplicationProcessSerialNumberHigh"`	The high long of the process serial number (PSN), as an `NSNumber` object.
`@"NSApplicationProcessSerialNumberLow"`	The low long of the process serial number (PSN), as an `NSNumber` object.

File Types

These constants specify different types of files returned by `getInfoForFile:application:type:` (page 15).

```
NSString *NSPlainFileType;
NSString *NSDirectoryFileType;
NSString *NSApplicationFileType;
NSString *NSFilesystemFileType;
NSString *NSShellCommandFileType;
```

Constants

`NSPlainFileType`

Plain (untyped) file

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSDirectoryFileType`

Directory

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSApplicationFileType`

Cocoa application

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSFilesystemFileType`
File-system mount point

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSShellCommandFileType`
Executable shell command

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

Declared In
`NSWorkspace.h`

File Operation Constants

These constants specify different types of file operations used by `performFileOperation:source:destination:files:tag:` (page 29).

```
NSString *NSWorkspaceMoveOperation;
NSString *NSWorkspaceCopyOperation;
NSString *NSWorkspaceLinkOperation;
NSString *NSWorkspaceCompressOperation;
NSString *NSWorkspaceDecompressOperation;
NSString *NSWorkspaceEncryptOperation;
NSString *NSWorkspaceDecryptOperation;
NSString *NSWorkspaceDestroyOperation;
NSString *NSWorkspaceRecycleOperation;
NSString *NSWorkspaceDuplicateOperation;
```

Constants
`NSWorkspaceMoveOperation`
Move file to destination. Behaves the same as `movePath:toPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceCopyOperation`
Copy file to destination. Behaves the same as `copyPath:toPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLinkOperation`
Create hard link to file in destination. Behaves the same as `linkPath:toPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceCompressOperation`
Compress file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDecompressOperation`
Decompress file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceEncryptOperation`
Encrypt file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDecryptOperation`
Decrypt file. This operation always returns an error.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDestroyOperation`
Destroy file. Behaves the same as `removeFileAtPath:handler:`.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceRecycleOperation`
Move file to trash. The file is moved to the trash folder on the volume containing the file using the same semantics as `NSWorkspaceMoveOperation`. If a file with the same name currently exists in the trash folder, the new file is renamed. If no trash folder exists on the volume containing the file, the operation fails.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceDuplicateOperation`
Duplicate file in source directory.

Available in Mac OS X v10.0 and later.

Declared in `NSWorkspace.h`.

Declared In
`NSWorkspace.h`

NSWorkspaceLaunchOptions

Constants representing different launch options.

```
typedef unsigned int NSWorkspaceLaunchOptions;
```

Discussion
See "Launch Options" (page 37) for possible values.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSWorkspace.h`

Launch Options

These constants define launch options you can pass to `launchAppWithBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifier:` (page 19) and `openURLs:withAppBundleIdentifier:options:additionalEventParamDescriptor:launchIdentifiers:` (page 28).

```
enum {
     NSWorkspaceLaunchAndPrint = 0x00000002,
     NSWorkspaceLaunchInhibitingBackgroundOnly = 0x00000080,
     NSWorkspaceLaunchWithoutAddingToRecents = 0x00000100,
     NSWorkspaceLaunchWithoutActivation = 0x00000200,
     NSWorkspaceLaunchAsync = 0x00010000,
     NSWorkspaceLaunchAllowingClassicStartup = 0x00020000,
     NSWorkspaceLaunchPreferringClassic = 0x00040000,
     NSWorkspaceLaunchNewInstance = 0x00080000,
     NSWorkspaceLaunchAndHide = 0x00100000,
     NSWorkspaceLaunchAndHideOthers = 0x00200000,
     NSWorkspaceLaunchDefault = NSWorkspaceLaunchAsync |
NSWorkspaceLaunchAllowingClassicStartup
};
```

Constants

`NSWorkspaceLaunchAndPrint`

Print items instead of opening them.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchInhibitingBackgroundOnly`

Causes launch to fail if the target is background-only.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchWithoutAddingToRecents`

Do not add the application or documents to the Recents menu.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchWithoutActivation`

Launch the application but do not bring it into the foreground.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAsync`

Launch the application and return the results asynchronously.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAllowingClassicStartup`

Start up the Classic compatibility environment, if it is required by the application.

Available in Mac OS X v10.3 and later.

Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchPreferringClassic`
: Force the application to launch in the Classic compatibility environment.

  Available in Mac OS X v10.3 and later.

  Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchNewInstance`
: Create a new instance of the application, even if one is already running.

  Available in Mac OS X v10.3 and later.

  Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAndHide`
: Tell the application to hide itself as soon as it has finished launching.

  Available in Mac OS X v10.3 and later.

  Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchAndHideOthers`
: Hide all applications except the newly launched one.

  Available in Mac OS X v10.3 and later.

  Declared in `NSWorkspace.h`.

`NSWorkspaceLaunchDefault`
: Launch the application asynchronously and launch it in the Classic environment, if required.

  Available in Mac OS X v10.3 and later.

  Declared in `NSWorkspace.h`.

Declared In
`NSWorkspace.h`

Workspace icon creation options

These constants describe the `NSWorkspaceIconCreationOptions` values used by `setIcon:forFile:options:` (page 31).

```
enum {
    NSExcludeQuickDrawElementsIconCreationOption    = 1 << 1,
    NSExclude10_4ElementsIconCreationOption        = 1 << 2
};
```

Constants
`NSExcludeQuickDrawElementsIconCreationOption`
: Suppress generation of the QuickDraw format icon representations that are used Mac OS X v10.0 through v10.4.

  Available in Mac OS X v10.4 and later.

  Declared in `NSWorkspace.h`.

`NSExclude10_4ElementsIconCreationOption`
: Suppress generation of the new higher resolution icon representations that are supported in Mac OS X v10.4.

  Available in Mac OS X v10.4 and later.

  Declared in `NSWorkspace.h`.

Discussion
You can combine these using the C bitwise OR operator.

Declared In
NSWorkspace.h

Notifications

All NSWorkspace notifications are posted to the NSWorkspace object's own notification center, not the application's default notification center. Access this center using the NSWorkspace object's notificationCenter (page 24) method.

NSWorkspaceDidLaunchApplicationNotification

Posted when a new application has started up.

The notification object is the shared NSWorkspace instance. The *userInfo* dictionary contains the keys and values described in "Constants" (page 34).

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

Declared In
NSWorkspace.h

NSWorkspaceDidMountNotification

Posted when a new device has been mounted.

The notification object is the shared NSWorkspace instance. The *userInfo* dictionary contains the following information:

Key	Value
@"NSDevicePath"	The path where the device was mounted, as a string.

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

Declared In
NSWorkspace.h

NSWorkspaceDidPerformFileOperationNotification

Posted when a file operation has been performed in the receiving application.

The notification object is the shared NSWorkspace instance. The *userInfo* dictionary contains the following information:

Key	Value
@"NSOperationNumber"	An NSNumber object containing an integer indicating the type of file operation completed.

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

Declared In
`NSWorkspace.h`

NSWorkspaceDidTerminateApplicationNotification

Posted when an application finishes executing.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains the keys and values described in "Constants" (page 34).

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

Declared In
`NSWorkspace.h`

NSWorkspaceDidWakeNotification

Posted when the machine wakes from sleep.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceDidUnmountNotification

Posted when the Finder has unmounted a device.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSDevicePath"`	The path where the device was previously mounted, as an `NSString` object.

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

Declared In
`NSWorkspace.h`

NSWorkspaceSessionDidBecomeActiveNotification

Posted after a user session is switched in. This allows an application to re-enable some processing when a switched out session gets switched back in, for example.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceSessionDidResignActiveNotification

Posted before a user session is switched out. This allows an application to disable some processing when its user session is switched out, and re-enable when that session gets switched back in, for example.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

If an application is launched in an inactive session, `NSWorkspaceSessionDidResignActiveNotification` is sent after `NSApplicationWillFinishLaunchingNotification` and before sending `NSApplicationDidFinishLaunchingNotification`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSWorkspace.h`

NSWorkspaceWillLaunchApplicationNotification

Posted when the Finder is about to launch an application.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains the keys and values described in "Constants" (page 34).

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

Declared In
`NSWorkspace.h`

NSWorkspaceWillPowerOffNotification

Posted when the user has requested a logout or that the machine be powered off.

The notification object is the shared `NSWorkspace` instance. This notification does not contain a *`userInfo`* dictionary.

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

Declared In
NSWorkspace.h

NSWorkspaceWillSleepNotification

Posted before the machine goes to sleep. An observer of this message can delay sleep for up to 30 seconds while handling this notification.

The notification object is the shared `NSWorkspace` instance. The notification does not contain a *`userInfo`* dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
NSWorkspace.h

NSWorkspaceWillUnmountNotification

Posted when the Finder is about to unmount a device. This notification will not be delivered if a volume was forcibly and immediately made unavailable, such as when a FireWire drive is simply unplugged, because there is no chance to deliver it before the volume becomes unavailable.

The notification object is the shared `NSWorkspace` instance. The *`userInfo`* dictionary contains the following information:

Key	Value
`@"NSDevicePath"`	The path where the device is mounted, as a string.

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

Declared In
NSWorkspace.h

Document Revision History

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

Date	Notes
2007-04-27	Added Uniform Type Identifier methods.
2007-04-03	Clarified return value of absolutePathForAppBundleWithIdentifier: when the bundle does not exist.
2006-05-23	Added the return type of launchAppWithBundleIdentifier:options: additionalEventParamDescriptor:launchIdentifier: .
	First publication of this content as a separate document.

Document Revision History

Index

O

P

S

T

U

W