NSWorkspace

Inherits From: NSObject

Conforms To: NSObject (NSObject)

Declared In: AppKit/NSWorkspace.h

Class Description

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
- miscellaneous services such as animating an image and requesting additional time before power off

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

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

Note: On the Microsoft Windows platform, some of the methods in this class have no effect. Refer to the method descriptions below.

Method Types

Accessing the shared NSWorkspace	+ sharedWorkspace
Accessing the NSWorkspace notification center	– notificationCenter
Opening files	– openFile: – openFile:withApplication: – openFile:fromImage:at:inView: – openFile:withApplication:andDeactivate: – openTempFile:
Manipulating applications	– launchApplication: – launchApplication:showIcon:autolaunch: – hideOtherApplications

Manipulating files	– performFileOperation:source:destination:files:tag: – selectFile:inFileViewerRootedAtPath:
Requesting information about files	– iconForFile: – iconForFileType: – iconForFiles: – getInfoForFile:application:type: – fullPathForApplication: – getFileSystemInfoForPath:isRemovable:isWritable: isUnmountable:description:type:
Requesting additional time before logout	– extendPowerOffBy:
Tracking changes to the file system	– noteFileSystemChanged – fileSystemChanged
Updating registered services and file types	– findApplications
Tracking changes to the defaults database	– noteUserDefaultsChanged – userDefaultsChanged
Tracking status changes for applications and devices	– mountedRemovableMedia – mountNewRemovableMedia – checkForRemovableMedia
Animating an image	– slideImage:from:to:
Unmounting a device	– unmountAndEjectDeviceAtPath:

Class Methods

sharedWorkspace

+ (NSWorkspace *)**sharedWorkspace**

Returns the shared NSWorkspace instance.

Instance Methods

checkForRemovableMedia

– (void)**checkForRemovableMedia**

On the Mach platform, polls the system's drives for any disks that have been inserted but not yet mounted. **checkForRemovableMedia** doesn't wait until such disks are mounted; instead, it requests that the disk be mounted asynchronously and returns immediately.

This method has no effect on the Microsoft Windows platform.

See also: – **mountNewRemovableMedia**, – **mountedRemovableMedia**

extendPowerOffBy:

– (int)**extendPowerOffBy:**(int)*requested*

Requests *requested* milliseconds more time before the power goes off or the user logs out. Returns the number of additional milliseconds granted. On some platforms you might not be able to extend the time.

fileSystemChanged

– (BOOL)**fileSystemChanged**

On the Mach platform, returns YES if a change to the file system has been registered with a **noteFileSystemChanged** message since the last **fileSystemChanged** message; NO otherwise.

This method is not implemented on the Microsoft Windows platform. If you try to use it, it raises an NSInvalidArgumentException.

findApplications

– (void)**findApplications**

On the Mach platform, examines all applications in the normal places (**/LocalApps**, **/NextApps**, **/NextDeveloper/Apps**) and updates the records of registered services and file types.

This method has no effect on the Microsoft Windows platform.

fullPathForApplication:

– (NSString *)**fullPathForApplication:**(NSString *)*appName*

Returns the full path for the application *appName*, or **nil** if *appName* isn't in one of the normal places.

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

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

On the Mach platform, describes the file system at *fullPath*. This method has no effect on the Microsoft Windows platform.

Returns YES if *fullPath* is a file system mount point, NO otherwise. If the return value is YES, *description* describes the file system; this value can be used in strings, but it shouldn't be depended upon by program logic. Example values for description are "hard", "nfs", and "foreign". *fileSystemType* indicates the file system type; values could be "NeXT", "DOS", or other values. *removableFlag* is YES if the file system is on removable media, NO otherwise. *writableFlag* is YES if the file system's media is writable, NO otherwise. *unmountableFlag* returns YES if the file system is unmountable, NO otherwise.

getInfoForFile:application:type:

– (BOOL)**getInfoForFile:**(NSString *)*fullPath*
application:(NSString **)*appName*
type:(NSString **)*type*

Retrieves information about the file specified by *fullPath*. If this method returns YES, the NSString pointed to by *appName* is set to the application the system would use to open *fullPath*. The NSString pointed to by *type* contains one of the following values or a file name extension such as "rtf" indicating the file's type:

Value	Type of File
NSPlainFileType	Plain (untyped) file
NSDirectoryFileType	Directory
NSApplicationFileType	OpenStep application
NSFilesystemFileType	File system mount point
NSShellCommandFileType	Executable shell command

This method returns NO if it could not find *fullPath*.

See also: – **iconForFile:**, – **iconForFiles:**

hideOtherApplications

– (void)**hideOtherApplications**

Hides all applications other than the sender. This method has no effect on the Microsoft Windows platform. On the Mach platform, the user can hide all applications except the current one by Command-double-clicking an application's tile, so programmatic invocation of this method is usually unnecessary.

iconForFile:

– (NSImage *)**iconForFile:**(NSString *)*fullPath*

Returns an NSImage with the icon for the single file specified by *fullPath*.

See also: – **getInfoForFile:application:type:**, – **iconForFileType:**, – **iconForFiles:**

iconForFileType:

– (NSImage *)**iconForFileType:**(NSString *)*fileType*

Returns an NSImage the icon for the file type specified by *fileType*.

See also: – **iconForFile:**, – **iconForFiles:**

iconForFiles:

– (NSImage *)**iconForFiles:**(NSArray *)*fullPaths*

Returns an NSImage with the icon for the files specified in *fullPaths*, an array of NSStrings. If *fullPaths* specifies one file, its icon is returned. If *fullPaths* specifies more than one file, an icon representing the multiple selection is returned.

See also: – **iconForFile:**, – **iconForFileType:**

launchApplication:

– (BOOL)**launchApplication:**(NSString *)*appName*

Launches the application *appName*. *appName* need not be specified with a full path and, in the case of an application wrapper, can be specified with or without the **.app** extension. Returns YES if the application is successfully launched or already running, NO if it can't be launched.

Before this method begins, it posts an NSWorkspaceWillLaunchApplicationNotification to the NSWorkspace's notification center. When the operation is complete, it posts an NSWorkspaceDidLaunchApplicationNotification.

See also: – **launchApplication:showIcon:autolaunch:**

launchApplication:showIcon:autolaunch:

– (BOOL)**launchApplication:**(NSString *)*appName*
showIcon:(BOOL)*showIcon*
autolaunch:(BOOL)*autolaunch*

Launches the application *appName*. If *showIcon* is NO, the application's icon won't be placed on the screen. (The icon still exists, though.) If *autolaunch* is YES, the autolaunch default will be set as though the application were autolaunched at startup. This method is provided to enable daemon-like applications that lack a normal user interface and for use by alternative dock programs. 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 to the NSWorkspace's notification center. When the operation is complete, it posts an NSWorkspaceDidLaunchApplicationNotification.

See also: – **launchApplication:**

mountNewRemovableMedia

– (NSArray *)**mountNewRemovableMedia**

On the Mach platform, polls the system's drives for any disks that have been inserted but not yet mounted, waits until the new disks have been mounted, and returns an NSArray of NSStrings containing full pathnames to all newly mounted disks. This method posts an NSWorkspaceDidMountNotification to the NSWorkspace's notification center when it is finished.

This method is not implemented on the Microsoft Windows platform. If you try to use it, it raises an NSInvalidArgumentException.

See also: – **checkForRemovableMedia**, – **mountedRemovableMedia**

mountedRemovableMedia

– (NSArray *)**mountedRemovableMedia**

Returns an NSArray of NSStrings containing the full pathnames of all currently mounted removable disks. This method is not implemented on the Microsoft Windows platform. If you try to use it, it raises an NSInvalidArgumentException.

On the Mach platform, if the computer provides an interrupt or other notification when the user inserts a disk into a drive, the Workspace Manager will mount the disk immediately. However, if no notification is given, the Workspace Manager won't be aware that a disk needs to be mounted. On such systems, an application should invoke either **mountNewRemovableMedia:** or **checkForRemovableMedia** before invoking **mountedRemovableMedia:**. Either of these methods cause the Workspace Manager 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 Workspace Manager to mount it.

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

See also: – **checkForRemovableMedia**, – **mountNewRemovableMedia**

noteFileSystemChanged

– (void)**noteFileSystemChanged**

On the Mach platform, informs NSWorkspace that the file system has changed. NSWorkspace 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.

This method has no effect on the Microsoft Windows platform.

See also: – **fileSystemChanged**

noteUserDefaultsChanged

– (void)**noteUserDefaultsChanged**

On the Mach platform, informs NSWorkspace that the defaults database has changed. NSWorkspace then reads all the defaults it is interested in and reconfigures itself appropriately. For example, on Mach platforms, this method is used by the Preferences application to notify Workspace Manager whether the user prefers to see hidden files.

This method has no effect on the Microsoft Windows platform.

See also: – **userDefaultsChanged**

notificationCenter

– (NSNotificationCenter *)**notificationCenter**

Returns the notification center for workspace notifications.

openFile:

– (BOOL)**openFile:**(NSString *)*fullPath*

Opens the file specified by *fullPath* using the default application for its type; returns YES if file was successfully opened, NO otherwise. The sending application is deactivated before the request is sent.

See also: – **openFile:fromImage:at:inView:**, – **openFile:withApplication:**, – **openFile:withApplication:andDeactivate:**, – **openTempFile:**

openFile:fromImage:at:inView:

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

Opens the file specified by *fullPath* using the default application for its type. On the Mach platform, Workspace Manager provides 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 *aView*'s coordinates. On the Microsoft Windows platform, this method is the same as the **openFile:** method.

The sending application is deactivated before the request is sent. Returns YES if the file is successfully opened, NO otherwise.

See also: – **openFile:**, – **openFile:withApplication:**, – **openFile:withApplication:andDeactivate:**, – **openTempFile:**

openFile:withApplication:

– (BOOL)**openFile:**(NSString *)*fullPath* **withApplication:**(NSString *)*appName*

Opens the file specified by *fullPath* using the *appName* application. *appName* need not be specified with a full path and, in the case of an application wrapper, can be specified with or without the **.app** extension. The sending application is deactivated before the request is sent. Returns YES if the file is successfully opened, NO otherwise.

See also: – **openFile:**, – **openFile:withApplication:andDeactivate:**

openFile:withApplication:andDeactivate:

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

Opens the file specified by *fullPath* using the *appName* application. *appName* need not be specified with a full path and, in the case of an application wrapper, can be specified with or without the **.app** extension. If *appName* is **nil**, the default application for the file's type is used. If *flag* is YES, the sending application is deactivated before the request is sent, allowing the opening application to become the active application. Returns YES if the file is successfully opened, NO otherwise.

See also: – **openFile:**, – **openFile:withApplication:**, – **application:openFile:** (NSApplication delegate method)

openTempFile:

– (BOOL)**openTempFile:**(NSString *)*fullPath*

Opens the temporary file specified by *fullPath* using the default application for its type. 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. Returns YES if the file is successfully opened, NO otherwise.

See also: – **openFile:**, – **openFile:fromImage:at:inView:**, – **openFile:withApplication:**, – **openFile:withApplication:andDeactivate:**

performFileOperation:source:destination:files:tag:

– (BOOL)**performFileOperation:**(NSString *)*operation*
source:(NSString *)*source*
destination:(NSString *)*destination*
files:(NSArray *)*files*
tag:(int *)*tag*

Performs a file operation on a set of files in a particular directory. *operation* is some file operation, such as compressing or moving files. *files* contains NSString specifying the names of the files to be manipulated. The file names are given relative to the *source* directory. The list can contain both files and directories; all of them must be located directly within source (not in one of its subdirectories).

Some operations—such as moving, copying, and linking files—require a destination directory to be specified. If not, *destination* should be the empty string (@"").

The possible values for *operation* are:

Operation	Meaning
NSWorkspaceMoveOperation	Move file to destination
NSWorkspaceCopyOperation	Copy file to destination
NSWorkspaceLinkOperation	Create link to file in destination
NSWorkspaceCompressOperation	Compress file
NSWorkspaceDecompressOperation	Decompress file
NSWorkspaceEncryptOperation	Encrypt file
NSWorkspaceDecryptOperation	Decrypt file
NSWorkspaceDestroyOperation	Destroy file
NSWorkspaceRecycleOperation	Move file to recycler
NSWorkspaceDuplicateOperation	Duplicate file in source directory

Note: NSWorkspaceCompressOperation, NSWorkspaceDecompressOperation, NSWorkspaceEncryptOperation, and NSWorkspaceDecryptOperation are not available on the Microsoft Windows platform.

This method returns YES if the operation succeeded, NO otherwise, In *tag*, the method returns a negative integer if the operation fails, 0 if the operation is performed synchronously and succeeds, and a positive integer if the operation is performed asynchronously. The positive integer is a tag that identifies the requested file operation. Before this method returns, it posts an NSWorkspaceDidPerformFileOperationNotification to NSWorkspace's notification center.

selectFile:inFileViewerRootedAtPath:

– (BOOL)**selectFile:**(NSString *)*fullPath* **inFileViewerRootedAtPath:**(NSString *)*rootFullPath*

Selects the file specified by *fullPath*. If a path is specified by *rootFullPath*, a new file viewer is opened. If *rootFullPath* is an empty string (@""), the file is selected in the main viewer. Returns YES if the file is successfully selected, NO otherwise.

slideImage:from:to:

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

On Mach platforms, animates a sliding image of *image* from *fromPoint* to *toPoint*, specified in screen coordinates. This method has no effect on the Microsoft Windows platform.

unmountAndEjectDeviceAtPath:

– (BOOL)**unmountAndEjectDeviceAtPath:**(NSString *)*path*

Unmounts and ejects the device at path. Returns YES if unmount succeeded and NO otherwise. When this method begins, it posts an NSWorkspaceWillUnmountNotification to NSWorkspace's notification center. When it is finished, it posts an NSWorkspaceDidUnmountNotification.

userDefaultsChanged

– (BOOL)**userDefaultsChanged**

On the Mach platform, returns whether a change to the defaults database has been registered with a **noteUserDefaultsChanged** message since the last **userDefaultsChanged** message.

This method has no effect on the Microsoft Windows platform.

Notifications

All NSWorkspace notifications are posted to NSWorkspace's own notification center, not the application's default notification center. Access this center using the **notificationCenter** method.

```
NSNotificationCenter *workspaceCenter = [[NSWorkspace sharedWorkspace]
                                         notificationCenter];
```

NSWorkspaceDidLaunchApplicationNotification

Posted when a new application has started up.

The notification object contains:

Notification Object	the shared NSWorkspace instance
Userinfo	
Key	**Value**
NSApplicationName	The application being terminated

NSWorkspaceDidMountNotification

Posted when a new device has been mounted.

The notification object contains:

Notification Object	the shared NSWorkspace instance
Userinfo	
Key	**Value**
NSDevicePath	The path where the device was mounted

NSWorkspaceDidPerformFileOperationNotification

Posted when a file operation has been performed.

Notification Object	the shared NSWorkspace instance
Userinfo	
Key	**Value**
NSOperationNumber	A number indicating the type of file operation completed

NSWorkspaceDidTerminateApplicationNotification

Posted when an application finishes executing.

The notification object contains:

Notification Object	the shared NSWorkspace instance
Userinfo	
Key	**Value**
NSApplicationName	The application that terminated

NSWorkspaceDidUnmountNotification

Posted when the workspace has unmounted a device.

Notification Object	the shared NSWorkspace instance
Userinfo	
Key	**Value**
NSDevicePath	The path where the device was previously mounted

NSWorkspaceWillLaunchApplicationNotification

Posted when the workspace is about to launch an application.

Notification Object	the shared NSWorkspace instance
Userinfo	

Key	Value
NSApplicationName	The application about to be launched

NSWorkspaceWillPowerOffNotification

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

Notification Object	the shared NSWorkspace instance
Userinfo	None

NSWorkspaceWillUnmountNotification

Posted when the workspace is about to unmount a device.

Notification Object	the shared NSWorkspace instance
Userinfo	

Key	Value
NSDevicePath	The path where the device is mounted