INSIDE MACINTOSH

Folder Manager Reference

Updated for Mac OS 8.5

November 3, 1998
Technical Publications

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

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, 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 MANUAL, 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.

Folder Manager Reference

Contents

The Folder Manager allows you to find and search folders, create new folders, and control how files are routed between folders. Because you can use the Folder Manager to manipulate standard Mac OS folders without relying on their names, your program is tolerant of changes to folder names and easier to localize.

This document describes the Folder Manager programming interface. Portions of this document were previously released as part of *Mac OS 8 Toolbox Reference*. See Appendix A, “Version History” (page 37), for descriptions of changes to and prior releases of this document.

▲ **WARNING**
All Folder Manager functions may move or purge memory and cannot be called at interrupt time. ▲

Folder Manager Gestalt Selector

Before calling any Folder Manager functions, your application should pass the selector `gestaltFindFolderAttr` to the `Gestalt` function to determine which Folder Manager functions are available.

```
enum {
    gestaltFindFolderAttr   = 'fold'
};
```

Constant description

`gestaltFindFolderAttr`
The `Gestalt` selector passed to determine whether the Folder Manager is available. Produces a value whose bits you should test to determine what Folder Manager functionality is available:

```
enum {
    gestaltFindFolderPresent    = 0,
    gestaltFolderDescSupport    = 1
};
```

Constant descriptions

`gestaltFindFolderPresent`
: If this bit is set, the basic Folder Manager functionality defined by the functions `FindFolder` (page 7) and `ReleaseFolder` (page 9) is available. This bit is set for versions of the Mac OS prior to Mac OS 8.

`gestaltFolderDescSupport`
: If this bit is set, the extended Folder Manager functionality supporting folder descriptors and routings is available. This bit is set for versions of the Mac OS starting with Mac OS 8.

Folder Manager Functions

The Folder Manager provides functions that your application can use for

- "Manipulating Folders" (page 6). Read this section to learn how to find a folder and how to release the Trash folder.
- "Describing Folders" (page 10). Read this section to learn how to obtain and provide descriptive information about folders.
- "Routing Files" (page 17). Read this section to learn how to route files to particular folders.

Manipulating Folders

The Folder Manager provides the following functions for the manipulation of folders:

- `FindFolder` (page 7) obtains location information for system-related directories.
- `ReleaseFolder` (page 9) releases the Trash folder in preparation for the unmounting of a server volume.

FindFolder

Obtains location information for system-related directories.

```
pascal OSErr FindFolder (
                    short vRefNum,
                    OSType folderType,
                    Boolean createFolder,
                    short *foundVRefNum,
                    long *foundDirID);
```

`vRefNum` Pass the volume reference number (or the constant `kOnSystemDisk` for the startup disk) of the volume on which you want to locate a directory; see “Volume Constant” (page 34).

`folderType` Pass a four-character folder type, or a constant that represents the type, for the folder you want to find; see “Folder Type Constants” (page 26).

`createFolder` A value of type `Boolean`, as defined in “Create Folder Flag Constants” (page 24). Pass the constant `kCreateFolder` to create a directory if it does not already exist; otherwise, pass the constant `kDontCreateFolder`. Directories inside the System Folder are created only if the System Folder directory exists. The `FindFolder` function will not create a System Folder directory even if you specify the `kCreateFolder` constant in the `createFolder` parameter. Passing `kCreateFolder` will also not create a parent folder; if the parent of the target folder does not already exist, attempting to create the target will fail.

`foundVRefNum` A pointer to a value of type `short`. On return, the value specifies the volume reference number for the volume containing the directory specified in the `folderType` parameter.

`foundDirID` A pointer to a value of type `long`. On return, the value specifies the directory ID number for the directory specified in the `folderType` parameter.

function result A result code; see “Folder Manager Result Codes” (page 35). The result code `fnfErr` indicates that the type has not been found in the `'fld#'` resource, or the disk doesn’t have System Folder support, or the disk does not have desktop database

support for Desktop Folder—in all cases, the folder has not been found. The result code `dupFNErr` indicates that a file has been found instead of a folder.

DISCUSSION

As of Mac OS 8 and later, your application can add folders to the System Folder—or nest folders within other folders—and locate the folders via the `FindFolder` function. Prior to Mac OS 8, your application could only use `FindFolder` to find folders that were immediately inside of the System Folder, and a few other special folders such as the Trash folder and the System Folder itself. Now, once a folder (and any folders that it is nested within) is described in a folder descriptor—that is, registered using the function `AddFolderDescriptor` (page 10)—your application can use `FindFolder` to find the folder no matter where it is located.

Those folders you're most likely to want to access are Preferences and Trash. For example, you might wish to check for the existence of a user's configuration file in Preferences or, if your application runs out of disk storage when trying to save a file, check how much disk storage is taken by items in the Trash directory and report this to the user.

Note that the specified folder used for a given volume might be located on a different volume in future versions of system software; therefore, do not assume the volume that you specify in `vRefNum` and the volume returned through `foundVRefNum` will be the same.

SPECIAL CONSIDERATIONS

Prior to Mac OS 8, the Finder identified the subdirectories of the System Folder, and their folder types, in a resource of type `'fld#'` located in the System file. While some backwards compatibility support for `'fld#'` remains, it has been superseded by the `'nfd#'` resource under Mac OS 8 and later. As with `'fld#'`, you should not modify or rely on the contents of the `'nfd#'` resource in the System file. Instead, use only the `FindFolder` function to find the appropriate folders, and use the functions `AddFolderDescriptor` (page 10) and `RemoveFolderDescriptor` (page 16) to modify folder descriptors.

VERSION NOTES

Available under System 7 and later.

Changed in Mac OS 8 to support finding folders registered using the function `AddFolderDescriptor` (page 10).

ReleaseFolder

Releases the Trash folder in preparation for unmounting a server volume.

```
pascal OSErr ReleaseFolder (
                    short vRefNum,
                    OSType folderType);
```

`vRefNum`	Pass the volume reference number of the server volume on which you want to release the Trash folder.
`folderType`	Always pass the `kTrashFolderType` constant. Other folder types are currently ignored.
function result	A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

When you call `FindFolder` (page 7) with the `kTrashFolderType` constant, it opens a file on a server volume that ensures each server volume user gets a unique Trash folder. Because a server volume's Trash folder may contain files or folders put there by the user, applications should delete the contents of the server volume's Trash folder. To do this, before your application unmounts a server volume, your application should call `ReleaseFolder`, or the `UnmountVol` request could fail with a `fBsyErr` result code. `ReleaseFolder` closes the file `FindFolder` may have opened and releases the Trash folder on that volume.

IMPORTANT

Your application should not use this function unless you want to unmount one or more server volumes. Normally, applications should not unmount servers; they should let users use the Finder to unmount volumes. In particular, applications should have no need to release the Trash folder explicitly; rather, unmounting volumes should be left to users to do with the Finder or by restarting. ▲

VERSION NOTES

Available under System 7.1 and later.

Describing Folders

As of Mac OS 8 and later, your application can add folders to the System Folder—or nest folders within other folders—and use the Folder Manager to find the folders. Once a folder (and any folders that it is nested within) is described in a **folder descriptor**, your application can use the Folder Manager to find the folder no matter where it is located.

The Folder Manager provides the following functions for describing folders:

- `AddFolderDescriptor` (page 10) copies the supplied information into a new folder descriptor entry in the system folder list.
- `GetFolderDescriptor` (page 12) obtains the folder descriptor information for the specified folder type from the global descriptor list.
- `GetFolderName` (page 13) obtains the name of the specified folder.
- `GetFolderTypes` (page 14) obtains the folder types contained in the global descriptor list.
- `IdentifyFolder` (page 15) obtains the folder type for the specified folder.
- `InvalidateFolderDescriptorCache` (page 15) invalidates any prior `FindFolder` results for the specified folder.
- `RemoveFolderDescriptor` (page 16) deletes the specified folder descriptor entry from the system folder list.

AddFolderDescriptor

Copies the supplied information into a new folder descriptor entry in the system folder list.

```
pascal OSErr AddFolderDescriptor (
                    FolderType foldType,
                    FolderDescFlags flags,
                    FolderClass foldClass,
                    FolderLocation foldLocation,
                    OSType badgeSignature,
```

```
                    OSType badgeType,
                    ConstStr63Param name,
                    Boolean replaceFlag);
```

`foldType` Pass a constant identifying the type of the folder you wish the Folder Manager to be able to find. See “Folder Type Constants” (page 26).

`flags` Set these flags to indicate whether a folder is created during startup, if the folder name is locked, and if the folder is created invisible; see “Folder Descriptor Flag Constants” (page 25).

`foldClass` Pass the class of the folder which you wish the Folder Manager to be able to find. The folder class determines how the `foldLocation` parameter is interpreted. See “Folder Descriptor Class Constants” (page 24) for a discussion of relative and special folder classes.

`foldLocation` For a relative folder, specify the folder type of the parent folder of the target. For a special folder, specify the location of the folder; see “Folder Descriptor Location Constants” (page 25).

`badgeSignature`
Reserved. Pass 0.

`badgeType` Reserved. Pass 0.

`name` A string specifying the name of the desired folder. For relative folders, this is the exact name of the desired folder. For special folders, the actual target folder may have a different name than the name specified in the folder descriptor. For example, the System Folder is often given a different name, but it can still be located with `FindFolder` (page 7).

`replaceFlag` Pass a `Boolean` value indicating whether you wish to replace a folder descriptor that already exists for the specified folder type. If `true`, it replaces the folder descriptor for the specified folder type. If `false`, it does not replace the folder descriptor for the specified folder type.

function result A result code; see “Folder Manager Result Codes” (page 35). The result code `duplicateFolderDescErr` indicates that a folder descriptor is already installed with the specified folder type and `replaceFlag` is false.

DISCUSSION

The `AddFolderDescriptor` function copies the supplied information into a new descriptor entry in the system folder list. You need to provide folder descriptors for each folder you wish the Folder Manager to be able to find via the function `FindFolder` (page 7). For example, a child folder located in a parent folder needs to have a descriptor created both for it and its parent folder, so that the child can be found.

VERSION NOTES

Supported under Mac OS 8 and later.

GetFolderDescriptor

Obtains the folder descriptor information for the specified folder type from the global descriptor list.

```
pascal OSErr GetFolderDescriptor (
                    FolderType foldType,
                    Size descSize,
                    FolderDesc *foldDesc);
```

`foldType` Pass a constant identifying the type of the folder for which you wish to get descriptor information. See “Folder Type Constants” (page 26).

`descSize` Pass the size (in bytes) of the folder descriptor structure for which a pointer is passed in the `foldDesc` parameter. This value is needed in order to determine the version of the structure being used.

`foldDesc` Pass a pointer to a folder descriptor structure. On return, the folder descriptor structure contains information from the global descriptor list for the specified folder type.

function result A result code; see “Folder Manager Result Codes” (page 35).

VERSION NOTES

Supported under Mac OS 8 and later.

GetFolderName

Obtains the name of the specified folder.

```
pascal OSErr GetFolderName (
                    short vRefNum,
                    OSType foldType,
                    short *foundVRefNum,
                    Str63 name);
```

`vRefNum`	Pass the volume reference number (or the constant `kOnSystemDisk` for the startup disk) of the volume containing the folder for which you wish the name to be identified.
`foldType`	Pass a constant identifying the type of the folder for which you wish the name to be identified. See “Folder Type Constants” (page 26).
`foundVRefNum`	A pointer to a value of type `short`. On return, the value is set to the volume reference number for the volume containing the folder specified in the `foldType` parameter.
`name`	On return, a string containing the title of the folder specified in the `foldType` and `vRefNum` parameters.
function result	A result code; see “Folder Manager Result Codes” (page 35).

DISCUSSION

The `GetFolderName` function obtains the name of the folder in the folder descriptor, not the name of the folder on the disk. The names may differ for a few special folders such as the System Folder. For relative folders, however, the actual name is always returned. You typically do not need to call this function.

VERSION NOTES

Supported under Mac OS 8 and later.

GetFolderTypes

Obtains the folder types contained in the global descriptor list.

```
pascal OSErr GetFolderTypes (
                    UInt32 requestedTypeCount,
                    UInt32 *totalTypeCount,
                    FolderType *theTypes);
```

`requestedTypeCount`
Pass the number of `FolderType` values that can fit in the buffer pointed to by the `theTypes` parameter; see “Folder Type Constants” (page 26).

`totalTypeCount`
Pass a pointer to an unsigned 32-bit integer value. On return, the value is set to the total number of `FolderType` values in the list. The `totalTypeCount` parameter may produce a value that is larger or smaller than that of the `requestedTypeCount` parameter. If `totalTypeCount` is equal to or smaller than the value passed in for `requestedTypeCount` and the value produced by the `theTypes` parameter is non-`nil`, then all folder types were returned to the caller.

`theTypes` Pass a pointer to an array of `FolderType` values; see “Folder Type Constants” (page 26). On return, the array contains the folder types for the installed descriptors. You can step through the array and call `GetFolderDescriptor` for each folder type. Pass `nil` if you only want to know the number of descriptors installed in the system’s global list, rather than the actual folder types of those descriptors.

function result A result code; see “Folder Manager Result Codes” (page 35).

VERSION NOTES

Supported under Mac OS 8 and later.

IdentifyFolder

Obtains the folder type for the specified folder.

```
pascal OSErr IdentifyFolder (
                    short vRefNum,
                    long dirID,
                    FolderType *foldType);
```

`vRefNum`	Pass the volume reference number (or the constant `kOnSystemDisk` for the startup disk) of the volume containing the folder whose type you wish to identify.
`dirID`	Pass the directory ID number for the folder whose type you wish to identify.
`foldType`	Pass a pointer to a value of type `FolderType`. On return, the value is set to the folder type of the folder with the specified `vRefNum` and `dirID` parameters; see "Folder Type Constants" (page 26) for descriptions of possible values.
function result	A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

The folder type is identified for the folder specified by the `vRefNum` and `dirID` parameters, if such a folder exists. Note that `IdentifyFolder` may take several seconds to complete. Note also that if there are multiple folder descriptors that map to an individual folder, `IdentifyFolder` returns the folder type of only the first matching descriptor that it finds.

VERSION NOTES

Supported under Mac OS 8 and later.

InvalidateFolderDescriptorCache

Invalidates any prior `FindFolder` results for the specified folder.

```
pascal OSErr InvalidateFolderDescriptorCache (
                    short vRefNum,
                    long dirID);
```

`vRefNum`	Pass the volume reference number (or the constant `kOnSystemDisk` for the startup disk) of the volume containing the folder for which you wish the descriptor cache to be invalidated. Pass 0 to completely invalidate all folder cache information.
`dirID`	Pass the directory ID number for the folder for which you wish the descriptor cache to be invalidated. Pass 0 to invalidate the cache for all folders on the specified disk.
function result	A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

The `InvalidateFolderDescriptorCache` function searches to see if there is currently a cache of results from `FindFolder` calls on the specified folder. If so, it invalidates the cache from the previous calls to the `FindFolder` function in order to force the Folder Manager to reexamine the disk when `FindFolder` is called again on the specified directory ID or volume reference number.

You should not normally need to call `InvalidateFolderDescriptorCache`.

VERSION NOTES

Supported under Mac OS 8 and later.

RemoveFolderDescriptor

Deletes the specified folder descriptor entry from the system folder list.

```
pascal OSErr RemoveFolderDescriptor (FolderType foldType);
```

`foldType`	Pass a constant identifying the type of the folder for which you wish to remove a descriptor. See "Folder Type Constants" (page 26).
function result	A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

Once a folder descriptor has been removed, the function `FindFolder` (page 7) will no longer be able to locate the folder type.

VERSION NOTES

Supported under Mac OS 8 and later.

Routing Files

With Mac OS 8.5 and later, your application can also use the Folder Manager to specify folder routings. **Folder routings** determine the folder to which the Finder routes a file of a given type.

The Folder Manager provides the following functions for routing files:

- `AddFolderRouting` (page 17) adds a folder routing structure to the global routing list.
- `FindFolderRouting` (page 18) finds the destination folder from a matching folder routing structure for the specified file.
- `GetFolderRoutings` (page 19) obtains folder routing information from the global routing list.
- `RemoveFolderRouting` (page 20) deletes a folder routing structure from the global routing list.

AddFolderRouting

Adds a folder routing structure to the global routing list.

```
pascal OSErr AddFolderRouting (
                    OSType fileType,
                    FolderType routeFromFolder,
                    FolderType routeToFolder,
                    RoutingFlags flags,
                    Boolean replaceFlag);
```

`fileType` Pass the `OSType` of the file to be routed.

`routeFromFolder`
Pass the folder type of the “from” folder; see “Folder Type Constants” (page 26) for descriptions of possible values. An item dropped on the folder specified in this parameter will be routed to the folder specified in the `routeToFolder` parameter.

`routeToFolder` The folder type of the "to" folder; see "Folder Type Constants" (page 26) for descriptions of possible values.

`flags` Reserved for future use; pass 0.

`replaceFlag` Pass a `Boolean` value indicating whether you wish to replace a folder routing that already exists. If `true`, it replaces the folder to which the item is being routed. If `false`, it leaves the folder to which the item is being routed.

function result A result code; see "Folder Manager Result Codes" (page 35). The result code `duplicateRoutingErr` indicates that a folder routing is already installed with the specified folder type and `replaceFlag` is false.

DISCUSSION

Your application can use the `AddFolderRouting` function to specify how the Finder routes a given file type.

VERSION NOTES

Supported under Mac OS 8.5 and later.

FindFolderRouting

Finds the destination folder from a matching folder routing structure for the specified file.

```
pascal OSErr FindFolderRouting (
                    OSType fileType,
                    FolderType routeFromFolder,
                    FolderType *routeToFolder,
                    RoutingFlags *flags);
```

`fileType` Pass the file type specified in the appropriate folder routing structure for the file for which you wish to find a destination folder.

`routeFromFolder`
Pass the folder type of the "from" folder for which you wish to find a "to" folder; see "Folder Type Constants" (page 26) for

descriptions of possible values. An item dropped on the folder specified in this parameter will be routed to the folder specified in the `routeToFolder` parameter.

`routeToFolder` A pointer to a value of type `FolderType`. On return, the value is set to the folder type of the destination folder. See "Folder Type Constants" (page 26) for descriptions of possible values

`flags` Reserved; pass 0.

function result A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

Both the file type and the folder type specified must match those of a folder routing structure in the global routing list for the `FindFolderRouting` function to succeed.

SPECIAL CONSIDERATIONS

The system initializes the Folder Manager's routing tables with a resource of type `'nrt#'` located in the System file. You should not modify or rely on the contents of the `'nrt#'` resource in the System file; use only the `FindFolderRouting` function to find the appropriate folder routing information.

VERSION NOTES

Supported under Mac OS 8 and later.

GetFolderRoutings

Obtains folder routing information from the global routing list.

```
pascal OSErr GetFolderRoutings (
                    UInt32 requestedRoutingCount,
                    UInt32 *totalRoutingCount,
                    Size routingSize,
                    FolderRouting *theRoutings);
```

`requestedRoutingCount`
An unsigned 32-bit value. Pass the number of folder routing structures that can fit in the buffer pointed to by the `theRoutings` parameter.

`totalRoutingCount`
A pointer to an unsigned 32-bit value. On return, the value is set to the number of folder routing structures in the global list. If this value is less than or equal to `requestedRoutingCount`, all folder routing structures were returned to the caller.

`routingSize` Pass the size (in bytes) of the `FolderRouting` structure.

`theRoutings` Pass a pointer to an array of `FolderRouting` (page 22) structures. On return the structure(s) contain the requested routing information. You may pass `nil` if you do not wish this information.

function result A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

The folder routing information in the global routing list determines how the Finder routes files.

VERSION NOTES

Supported under Mac OS 8 and later.

RemoveFolderRouting

Deletes a folder routing structure from the global routing list.

```
pascal OSErr RemoveFolderRouting (
                    OSType fileType,
                    FolderType routeFromFolder);
```

`fileType` Pass the file type value contained in the folder routing structure to be removed.

`routeFromFolder`
Pass the folder type of the "from" folder; see "Folder Type Constants" (page 26) for descriptions of possible values.

function result A result code; see "Folder Manager Result Codes" (page 35).

DISCUSSION

Both the file type and the folder type specified must match those of an existing folder routing structure in the global routing list for the `RemoveFolderRouting` function to succeed.

VERSION NOTES

Supported under Mac OS 8.5 and later.

Folder Manager Data Types

The Folder Manager provides the following data types:

- `FolderDesc` (page 21)
- `FolderRouting` (page 22)

FolderDesc

The `FolderDesc` structure can be used to find existing folder descriptors and create new ones; it is supported under Mac OS 8 and later.

```
struct FolderDesc {
    size                    descSize;
    FolderType              foldType;
    FolderDescFlags         flags;
    FolderClass             foldClass;
    FolderType              foldLocation;
    OSType                  badgeSignature;
    OSType                  badgeType;
    UInt32                  reserved;
```

```
    Str63                   name;
};
typedef struct FolderDesc FolderDesc;
typedef FolderDesc *FolderDescPtr;
```

Field descriptions

`descSize`	The size (in bytes) of this structure.
`foldType`	A constant of type `FolderType` that identifies the kind of target folder. See “Folder Type Constants” (page 26) for a list of possible folder types.
`flags`	Flags indicating whether a folder is created during startup, if the folder name is locked, and if the folder created is invisible; see “Folder Descriptor Flag Constants” (page 25).
`foldClass`	The class indicating whether the folder is relative to the parent folder or special; see “Folder Descriptor Class Constants” (page 24).
`foldLocation`	For a relative folder, the `foldLocation` field specifies the `FolderType` of the parent folder of the target. For special folders, the location of the folder. See “Folder Descriptor Location Constants” (page 25).
`badgeSignature`	Reserved. Set this field to 0.
`badgeType`	Reserved. Set this field to 0.
`reserved`	Reserved. Set this field to 0.
`name`	A string specifying the name of the desired folder. For relative folders, this will be the exact name of the desired folder. For special folders, the actual target folder may have a different name than the name specified in the folder descriptor. For example, the System Folder is often given a different name, but it can still be located with `FindFolder` (page 7).

FolderRouting

The folder routing structure specifies the folder that files are routed to, based on the folder they are routed from. The `FolderRouting` structure is supported under Mac OS 8 and later.

```
struct FolderRouting {
    Size                descSize;
    OSType              fileType;
    FolderType          routeFromFolder;
    FolderType          routeToFolder;
    RoutingFlags        flags;
};
typedef struct FolderRouting FolderRouting;
typedef FolderRouting *FolderRoutingPtr;
```

Field descriptions

`descSize`	The size (in bytes) of this structure.
`fileType`	A constant of type `OSType` that describes the file type of the item to be routed.
`routeFromFolder`	The folder type identifying the folder from which an item will be routed. If an item is dropped on the folder specified in the `routeFromFolder` field, it will be routed to the folder described in the `routeToFolder` field. See "Folder Type Constants" (page 26) for a list of possible values.
`routeToFolder`	The folder type identifying the folder to which an item will be routed; see "Folder Type Constants" (page 26) for a list of possible values.
`flags`	Reserved. Set this field to 0.

Folder Manager Constants

The Folder Manager provides the following constants:

- "Create Folder Flag Constants" (page 24)
- "Folder Descriptor Class Constants" (page 24)
- "Folder Descriptor Flag Constants" (page 25)
- "Folder Descriptor Location Constants" (page 25)
- "Folder Type Constants" (page 26)
- "Volume Constant" (page 34)

Create Folder Flag Constants

You can pass these flag constants in the `createFolder` parameter of the function `FindFolder` (page 7) to indicate whether a folder should be created, if it is not found.

```
enum {
    kCreateFolder       = true,
    kDontCreateFolder   = false
};
```

Constant descriptions

`kCreateFolder` Specifies that the folder should be created, if it is not found.

`kDontCreateFolder` Specifies that the folder should not be created, if it is not found.

Folder Descriptor Class Constants

Constants of type `FolderClass` are used to specify how folder location information should be interpreted in the function `AddFolderDescriptor` (page 10) and the structure `FolderDesc` (page 21). The `FolderClass` constants are supported under Mac OS 8 and later.

IMPORTANT

Developers can only create new folder descriptors with a class of `kRelativeFolder`.

```
enum {
    kRelativeFolder = 'relf',
    kSpecialFolder  = 'spcf'
};
typedef OSType FolderClass;
```

Constant descriptions

`kRelativeFolder` Relative folders are located in terms of the folders in which they are nested, that is, their parent folders. This constant indicates that the folder location specified is the folder type of the parent folder, and the name specified is the name of the folder. Most folder descriptors are for relative folders.

`kSpecialFolder` Special folders—such as the System Folder and the disk's root directory—are in set locations that are not determined relative to any other folder. This constant indicates that the folder is located algorithmically, according to the constant supplied for the folder location (`kBlessedFolder` or `kRootFolder`). Developers cannot create new folder descriptors of the `kSpecialFolder` class.

Folder Descriptor Flag Constants

The `FolderDescFlags` enumeration defines masks your application can use in the `AddFolderDescriptor` (page 10) function and the `FolderDesc` (page 21) structure to specify various attributes of a folder. All other flag bits are reserved for future use. The `FolderDescFlags` constants are supported under Mac OS 8 and later.

You may set any combination of the following bits:

```
enum {
    kCreateFolderAtBoot         = 0x00000002,
    kFolderCreatedInvisible     = 0x00000004,
    kFolderCreatedNameLocked    = 0x00000008
};
typedef UInt32 FolderDescFlags;
```

Constant descriptions

`kCreateFolderAtBoot`
If the bit specified by this mask is set, the folder is created during startup if needed.

`kFolderCreatedInvisible`
If the bit specified by this mask is set, the folder created is invisible.

`kFolderCreatedNameLocked`
If the bit specified by this mask is set, the name of the folder is locked when the folder is created.

Folder Descriptor Location Constants

There are two special folder locations that you can specify in folder descriptors via the `FolderDesc` (page 21) structure and the `AddFolderDescriptor` (page 10)

function. For folders whose class is `kSpecialFolder`, you can use the following constants to specify the location of the folder algorithmically. The `FolderLocation` constants are supported under Mac OS 8 and later.

```
enum {
    kBlessedFolder      = 'blsf',
    kRootFolder         = 'rotf'
};
typedef OSType FolderType;
typedef OSType FolderLocation;
```

Constant descriptions

`kBlessedFolder`	Indicates that the folder location is the System Folder on the volume.
`kRootFolder`	Indicates that the folder location is the root directory of the volume.

Folder Type Constants

You can use `folderType` constants to specify a type of folder on a particular volume.

```
enum {
    kSystemFolderType                   = 'macs',
    kDesktopFolderType                  = 'desk',
    kTrashFolderType                    = 'trsh',
    kWhereToEmptyTrashFolderType        = 'empt',
    kPrintMonitorDocsFolderType         = 'prnt',
    kStartupFolderType                  = 'strt',
    kShutdownFolderType                 = 'shdf',
    kFontsFolderType                    = 'font',
    kAppleMenuFolderType                = 'amnu',
    kControlPanelFolderType             = 'ctrl',
    kExtensionFolderType                = 'extn',
    kPreferencesFolderType              = 'pref',
    kTemporaryFolderType                = 'temp',
    kExtensionDisabledFolderType        = 'extD',
    kControlPanelDisabledFolderType     = 'ctrD',
    kSystemExtensionDisabledFolderType  = 'macD',
    kStartupItemsDisabledFolderType     = 'strD',
```

```
    kShutdownItemsDisabledFolderType    = 'shdD',
    kApplicationsFolderType             = 'apps',
    kDocumentsFolderType                = 'docs',
    kVolumeRootFolderType               = 'root',
    kChewableItemsFolderType            = 'flnt',
    kApplicationSupportFolderType       = 'asup',
    kTextEncodingsFolderType            = 'ftex',
    kStationeryFolderType               = 'odst',
    kOpenDocFolderType                  = 'odod',
    kOpenDocShellPlugInsFolderType      = 'odsp',
    kEditorsFolderType                  = 'oded',
    kOpenDocEditorsFolderType           = 'fodf',
    kOpenDocLibrariesFolderType         = 'odlb',
    kGenEditorsFolderType               = 'fedi',
    kHelpFolderType                     = 'fhlp',
    kInternetPlugInFolderType           = 'fnet',
    kModemScriptsFolderType             = 'fmod',
    kPrinterDescriptionFolderType       = 'ppdf',
    kPrinterDriverFolderType            = 'fprd',
    kScriptingAdditionsFolderType       = 'fscr',
    kSharedLibrariesFolderType          = 'flib',
    kVoicesFolderType                   = 'fvoc',
    kControlStripModulesFolderType      = 'sdev',
    kAssistantsFolderType               = 'astf',
    kUtilitiesFolderType                = 'utif',
    kAppleExtrasFolderType              = 'aexf',
    kContextualMenuItemsFolderType      = 'cmnu',
    kMacOSReadMesFolderType             = 'morf',
    kALMModulesFolderType               = 'walk',
    kALMPreferencesFolderType           = 'trip',
    kALMLocationsFolderType             = 'fall',
    kColorSyncProfilesFolderType        = 'prof',
    kThemesFolderType                   = 'thme',
    kFavoritesFolderType                = 'favs',
    kInternetFolderType                 = 'intf',
    kAppearanceFolderType               = 'appr',
    kSoundSetsFolderType                = 'snds',
    kDesktopPicturesFolderType          = 'dtpf',
    kInternetSearchSitesFolderType      = 'issf',
    kFindSupportFolderType              = 'fnds',
    kFindByContentFolderType            = 'fbcf',
```

```
    kInstallerLogsFolderType            = 'ilgf',
    kScriptsFolderType                  = 'scrf',
    kFolderActionsFolderType            = 'fasf',
    kLauncherItemsFolderType            = 'laun',
    kRecentApplicationsFolderType       = 'rapp',
    kRecentDocumentsFolderType          = 'rdoc',
    kRecentServersFolderType            = 'rsvr',
    kSpeakableItemsFolderType           = 'spki'
};
typedef OSType FolderType;
```

Constant descriptions

`kSystemFolderType` Specifies the System Folder.

`kDesktopFolderType` Specifies the Desktop Folder.

`kTrashFolderType` Specifies the single-user Trash folder.

`kWhereToEmptyTrashFolderType`
Specifies the shared Trash folder; on a file server, this indicates the parent directory of all logged-on users' Trash subdirectories.

`kPrintMonitorDocsFolderType`
Specifies the PrintMonitor Documents folder in the System Folder.

`kStartupFolderType`
Specifies the Startup Items folder in the System Folder.

`kShutdownFolderType`
Specifies the Shutdown Items folder in the System Folder.

`kAppleMenuFolderType`
Specifies the Apple Menu Items folder in the System Folder.

`kControlPanelFolderType`
Specifies the Control Panels folder in the System Folder.

`kExtensionFolderType`
Specifies the Extensions folder in the System Folder.

`kFontsFolderType` Specifies the Fonts folder in the System Folder.

`kPreferencesFolderType`
Specifies the Preferences folder in the System Folder.

`kTemporaryFolderType`
: Specifies the Temporary folder. This folder exists as an invisible folder at the volume root.

`kExtensionDisabledFolderType`
: Specifies the Extensions (Disabled) folder in the System Folder. Supported with Mac OS 8 and later.

`kControlPanelDisabledFolderType`
: Specifies the Control Panels (Disabled) folder in the System Folder. Supported with Mac OS 8 and later.

`kSystemExtensionDisabledFolderType`
: Specifies the System Extensions (Disabled) folder in the System Folder. Supported with Mac OS 8 and later.

`kStartupItemsDisabledFolderType`
: Specifies the Startup Items (Disabled) folder in the System Folder. Supported with Mac OS 8 and later.

`kShutdownItemsDisabledFolderType`
: Specifies the Shutdown Items (Disabled) folder in the System Folder. Supported with Mac OS 8 and later.

`kApplicationsFolderType`
: Specifies the Applications folder installed at the root level of the volume. Supported with Mac OS 8 and later.

`kDocumentsFolderType`
: Specifies the Documents folder. This folder is created at the volume root. Supported with Mac OS 8 and later.

`kVolumeRootFolderType`
: Specifies the root folder of a volume. Supported with Mac OS 8 and later.

`kChewableItemsFolderType`
: Specifies the invisible folder on the system disk called “Cleanup at Startup” whose contents are deleted when the system is restarted, instead of merely being moved to the Trash. When the `FindFolder` function indicates this folder is available (by returning `noErr`), developers should usually use this folder for their temporary items, in preference to the Temporary Folder. Supported with Mac OS 8 and later.

`kApplicationSupportFolderType`
: Specifies the Application Support folder in the System Folder. This folder contains code and data files needed by

third-party applications. These files should usually not be written to after they are installed. In general, files deleted from this folder remove functionality from an application, unlike files in the Preferences folder, which should be non-essential. One type of file that could be placed here would be plug-ins that the user might want to maintain separately from any application, such as for an image-processing application that has many “fourth-party” plug-ins that the user might want to upgrade separately from the host application. Another type of file that might belong in this folder would be application-specific data files that are not preferences, such as for a scanner application that needs to read description files for specific scanner models according to which are currently available on the SCSI bus or network. Supported with Mac OS 8 and later.

`kTextEncodingsFolderType`

Specifies the Text Encodings folder in the System Folder. Supported with Mac OS 8 and later.

`kStationeryFolderType`

Specifies the OpenDoc stationery folder. Supported with Mac OS 8 and later.

`kOpenDocFolderType`

Specifies the OpenDoc root folder. Supported with Mac OS 8 and later.

`kOpenDocShellPlugInsFolderType`

Specifies the OpenDoc shell plug-ins folder in the OpenDoc folder. Supported with Mac OS 8 and later.

`kEditorsFolderType`

Specifies the OpenDoc editors folder in the Mac OS folder. Supported with Mac OS 8 and later.

`kOpenDocEditorsFolderType`

Specifies the OpenDoc subfolder in the Editors folder. Supported with Mac OS 8 and later.

`kOpenDocLibrariesFolderType`

Specifies the OpenDoc libraries folder. Supported with Mac OS 8 and later.

`kGenEditorsFolderType`

Specifies a general editors folder. Supported with Mac OS 8 and later.

`kHelpFolderType` Specifies the Help folder in the System Folder. Supported with Mac OS 8 and later.

`kInternetPlugInFolderType`
Specifies the Browser Plug-ins folder in the System Folder. Supported with Mac OS 8 and later.

`kModemScriptsFolderType`
Specifies the Modem Scripts folder in the Extensions folder. Supported with Mac OS 8 and later.

`kPrinterDescriptionFolderType`
Specifies the Printer Descriptions folder in the Extensions folder. Supported with Mac OS 8 and later.

`kPrinterDriverFolderType`
Specifies the printer drivers folder. This constant is not currently supported.

`kScriptingAdditionsFolderType`
Specifies the Scripting Additions folder in the System Folder. Supported with Mac OS 8 and later.

`kSharedLibrariesFolderType`
Specifies the general shared libraries folder. This constant is not currently supported.

`kVoicesFolderType` Specifies the Voices folder in the Extensions folder. Supported with Mac OS 8 and later.

`kControlStripModulesFolderType`
Specifies the Control Strip Modules folder in the System Folder. Supported with Mac OS 8 and later.

`kAssistantsFolderType`
Specifies the Assistants folder installed at the root level of the volume. Supported with Mac OS 8 and later.

`kUtilitiesFolderType`
Specifies the Utilities folder installed at the root level of the volume. Supported with Mac OS 8 and later.

`kAppleExtrasFolderType`
Specifies the Apple Extras folder installed at the root level of the volume. Supported with Mac OS 8 and later.

`kContextualMenuItemsFolderType`
Specifies the Contextual Menu Items folder in the System Folder. Supported with Mac OS 8 and later.

`kMacOSReadMesFolderType`
: Specifies the Mac OS Read Me Files folder installed at the root level of the volume. Supported with Mac OS 8 and later.

`kALMModulesFolderType`
: Specifies the Location Manager Modules folder in the Extensions Folder. Supported with Mac OS 8.1 and later.

`kALMPreferencesFolderType`
: Specifies the Location Manager Prefs folder in the Preferences folder. Supported with Mac OS 8.1 and later.

`kALMLocationsFolderType`
: Specifies the Locations folder in the Location Manager Prefs folder. Files containing configuration information for different locations are stored here. Supported with Mac OS 8.1 and later.

`kColorSyncProfilesFolderType`
: Specifies the ColorSync Profiles folder in the System Folder. Supported with Mac OS 8.1 and later.

`kThemesFolderType`
: Specifies the Theme Files folder in the Appearance folder. Supported with Mac OS 8.1 and later.

`kFavoritesFolderType`
: Specifies the Favorites folder in the System Folder. This folder is for storing Internet location files, aliases, and aliases to other frequently used items. Facilities for adding items into this folder are found in Contextual Menus, the Finder, Navigation Services, and others. Supported with Mac OS 8.1 and later.

`kInternetFolderType`
: Specifies the Internet folder installed at the root level of the volume. This folder is a location for saving Internet-related applications, resources, and tools. Supported with Mac OS 8.5 and later.

`kAppearanceFolderType`
: Specifies the Appearance folder in the System Folder. Supported with Mac OS 8.5 and later.

`kSoundSetsFolderType`
: Specifies the Sound Sets folder in the Appearance folder. Supported with Mac OS 8.5 and later.

`kDesktopPicturesFolderType`

Specifies the Desktop Pictures folder in the Appearance folder. This folder is used for storing desktop picture files. Files of type `'JPEG'` are auto-routed into this folder when dropped into the System Folder. Supported with Mac OS 8.5 and later.

`kInternetSearchSitesFolderType`

Specifies the Internet Search Sites folder in the System Folder. This folder contains Internet search site specification files used by the Find application when it accesses Internet search sites. Files of type `'issp'` are auto-routed to this folder. Supported with Mac OS 8.5 and later.

`kFindSupportFolderType`

Specifies the Find folder in the Extensions folder. This folder contains files used by the Find application. Supported with Mac OS 8.5 and later.

`kFindByContentFolderType`

Specifies the Find By Content folder installed at the root level of the volume. This folder is invisible and its use is private to Find By Content. Supported with Mac OS 8.5 and later.

`kInstallerLogsFolderType`

Specifies the Installer Logs folder installed at the root level of the volume. You can use this folder to save installer log files. Supported with Mac OS 8.5 and later.

`kScriptsFolderType`

Specifies the Scripts folder in the System Folder. This folder is for saving AppleScript scripts. Supported with Mac OS 8.5 and later.

`kFolderActionsFolderType`

Specifies the Folder Action Scripts folder in the Scripts folder. Supported with Mac OS 8.5 and later.

`kLauncherItemsFolderType`

Specifies the Launcher Items folder in the System Folder. Items in this folder appear in the Launcher control panel. Items included in folders with names beginning with a bullet (Option-8) character will appear as a separate panel

in the Launcher window. Supported with Mac OS 8.5 and later.

`kRecentApplicationsFolderType`
Specifies the Recent Applications folder in the Apple Menu Items folder. Apple Menu Items saves aliases to recent applications here. Supported with Mac OS 8.5 and later.

`kRecentDocumentsFolderType`
Specifies the Recent Documents folder in the Apple Menu Items folder. Apple Menu Items saves aliases to recently opened documents here. Supported with Mac OS 8.5 and later.

`kRecentServersFolderType`
Specifies the Recent Servers folder in the Apple Menu Items folder. Apple Menu Items saves aliases to recently mounted servers here. Supported with Mac OS 8.5 and later.

`kSpeakableItemsFolderType`
Specifies the Speakable Items folder. This folder is for storing scripts and items recognized by speech recognition. Supported with Mac OS 8.5 and later.

Volume Constant

You can pass this constant in the `vRefNum` parameter of `FindFolder` (page 7) to locate a folder on the startup disk.

```
enum {
    kOnSystemDisk   = -32768L
};
```

Constant description

`kOnSystemDisk` Specifies the system disk.

Folder Manager Result Codes

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

`noErr`	0	No error
`nsvErr`	–35	Volume not found
`fnfErr`	–43	Folder not found
`dupFNErr`	–48	File found instead of folder
`dirNFErr`	–120	Parent directory not found
`badFolderDescErr`	–4270	Invalid folder
`duplicateFolderDescErr`	–4271	Duplicate folders for a particular routing
`invalidFolderTypeErr`	–4273	Invalid folder name
`duplicateRoutingErr`	–4274	Same routing for two folders
`routingNotFoundErr`	–4275	No routing set up for folder passed in
`badRoutingSizeErr`	–4276	Incorrect `descSize` field of the folder routing structure

APPENDIX A

Version History

This document has had the following releases:

Table A-1 *Folder Manager Reference* Revision History

Version	Notes
Nov. 3, 1998	First release of the *Folder Manager Reference* document. Portions of this document were previously included in *Mac OS 8 Toolbox Reference*. The following corrections and additions were made: "Folder Type Constants" (page 26). Added new folder constants for Mac OS 8.5. `AddFolderDescriptor` (page 10). Corrected function result information to indicate that `duplicateFolderDescErr` may be returned. `AddFolderRouting` (page 17). Described and noted support under Mac OS 8.5. `RemoveFolderRouting` (page 20). Described and noted support under Mac OS 8.5.
Jan. 15, 1998	The following corrections were made: "Folder Type Constants" (page 26). Amended the description of the `kApplicationSupportFolderType` constant.
Dec. 10, 1997	The following corrections were made: "Folder Type Constants" (page 26). Expanded the description of the `kApplicationSupportFolderType` constant.
Dec. 2, 1997	PDF formatting improved.
Nov. 3, 1997	First document release.

Index

N

R

U

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Adobe Letter Gothic.

WRITER
Donna S. Lee

PRODUCTION EDITOR
Glen Frank

Acknowledgments to Darren Litzinger and Greg Robbins.