Alias Manager Reference

Carbon > File Management

2006-12-05

Contents

Tables

Alias Manager Reference

Framework:	CoreServices/CoreServices.h
Declared in	Aliases.h

Overview

The Alias Manager creates and resolves alias records, which are data structures that describe file system objects (files, directories, and volumes.) An alias record contains a "fingerprint" of a file system object. You can store the alias record instead of a file system reference, and use the Alias Manager to find the object again when it's needed. The Alias Manager contains algorithms for locating objects that have been moved, renamed, copied, or restored from backup.

The exact makeup of an alias record depends on the file system in which the object resides. The Alias Manager takes advantage of persistent object ids, creation dates, file types, creator codes and the like if they are available. By default, an object at the location stored in the alias record will be considered a stronger match than an object with the same file id in a different location. (You can alter this behavior by passing flags to the functions that resolve the alias.)

The Alias Manager supports two types of alias records. The standard alias contains as much information as the Alias Manager can gather from the underlying file system. The minimal alias only stores a subset of the information in a standard alias record. A minimal alias may be used when the object is unlikely to move, the reference is to be short-lived, or space is a critical issue (the exact space savings depends on the underlying file system format.) The standard alias is the preferred format because it is more robust.

The Finder supports the creation and use of alias files that contain alias records. Currently, Mac OS X does not provide a way for other applications to create these alias files. The Alias Manager can identify and resolve Finder alias files, but it cannot create them.

Functions by Task

Creating and Updating Alias Records

`FSNewAlias` (page 15)
    Creates a new alias record, given a target file or directory.

`FSNewAliasUnicode` (page 18)
    Creates a new alias record, given the Unicode name and parent directory of the target.

`FSNewAliasFromPath` (page 16)
    Creates a new alias record, given the pathname of the target file or directory.

`FSNewAliasMinimal` (page 16)
: Creates a new minimal alias record, given a target file or directory.

`FSNewAliasMinimalUnicode` (page 17)
: Creates a minimal alias, given the Unicode name and parent directory of the target.

`FSUpdateAlias` (page 23)
: Updates an alias record for a specified target.

Getting Alias Size

`GetAliasSize` (page 24)
: Gets the size of an alias record referenced by a handle.

`GetAliasSizeFromPtr` (page 24)
: Gets the size of an alias record referenced by a pointer.

Getting and Setting Alias User Types

`GetAliasUserType` (page 24)
: Gets the user type for an alias record referenced by a handle.

`SetAliasUserType` (page 26)
: Sets the user type for an alias record referenced by a handle.

`GetAliasUserTypeFromPtr` (page 25)
: Gets the user type for the alias record referenced by a pointer.

`SetAliasUserTypeWithPtr` (page 27)
: Sets the user type for the alias record referenced by a pointer.

Resolving and Reading Alias Records

`FSCopyAliasInfo` (page 11)
: Returns information from an alias handle.

`FSMatchAliasBulk` (page 13)
: Identifies a list of possible matches for an alias.

`FSResolveAlias` (page 19)
: Returns an `FSRef` to the single most likely target of an alias record.

`FSResolveAliasWithMountFlags` (page 22)
: Returns an `FSRef` to the target of an alias.

`FSMatchAlias` (page 49) Deprecated in Mac OS X v10.5
: Identifies a list of possible matches for an alias. (Deprecated. Use `FSMatchAliasBulk` (page 13) instead.)

`FSMatchAliasNoUI` (page 50) Deprecated in Mac OS X v10.5
: Identifies a list of possible matches for an alias without any user interaction. (Deprecated. Use `FSMatchAliasBulk` (page 13) with the `kARMNoUI` flag instead.)

Working With Finder Alias Files

`FSFollowFinderAlias` (page 12)
: Resolves an alias record obtained from a Finder alias file.

`FSIsAliasFile` (page 12)
: Determines whether a file system object is an alias file, a data file, or a folder.

`FSResolveAliasFile` (page 20)
: Resolves an alias contained in an alias file.

`FSResolveAliasFileWithMountFlags` (page 21)
: Resolves an alias contained in an alias file.

Working With Universal Procedure Pointers to Alias Manager Callbacks

`NewAliasFilterUPP` (page 26)
: Creates a new universal procedure pointer (UPP) to an alias filtering callback function.

`DisposeAliasFilterUPP` (page 10)
: Disposes of a universal procedure pointer (UPP) to an alias filtering callback function.

`InvokeAliasFilterUPP` (page 25)
: Calls your alias filtering callback function.

Deprecated Functions

Alias Manager functions that use the `FSSpec` data type have been deprecated. Instead, you should use the equivalent `FSRef`–based functions, which include support for features such as unicode and long filenames. For more information on `FSSpec` and `FSRef` types, see *File Manager Reference*.

`GetAliasInfo` (page 37) Deprecated in Mac OS X v10.3
: Gets information from an alias record without actually resolving the record. (Deprecated. Use `FSCopyAliasInfo` (page 11) instead.)

`FollowFinderAlias` (page 48) Deprecated in Mac OS X v10.5
: Resolves an alias record obtained from a Finder alias file. (Deprecated. Use `FSFollowFinderAlias` (page 12) instead.)

`MatchAliasNoUI` (page 52) Deprecated in Mac OS X v10.5
: Identifies a list of possible matches for an alias without any user interaction. (Deprecated. Use `FSMatchAliasBulk` (page 13) with the `kARMNoUI` flag instead.)

`ResolveAliasFileWithMountFlags` (page 53) Deprecated in Mac OS X v10.5
: Resolves an alias contained in an alias file. (Deprecated. Use `FSResolveAliasFileWithMountFlags` (page 21) instead.)

`IsAliasFile` (page 38) Deprecated in Mac OS X v10.4
: Determines whether a file system object is an alias file, a data file, or a folder. (Deprecated. Use `FSIsAliasFile` (page 12) instead.)

`MatchAlias` (page 39) Deprecated in Mac OS X v10.4
: Identifies a list of possible matches for an alias and passes the list through an optional selection filter. The filter can return more than one possible match. (Deprecated. Use `FSMatchAliasBulk` (page 13) instead.)

`NewAlias` (page 40) Deprecated in Mac OS X v10.4
Creates a complete alias record. (Deprecated. Use `FSNewAlias` (page 15) instead.)

`NewAliasMinimal` (page 41) Deprecated in Mac OS X v10.4
Creates a short alias record quickly. (Deprecated. Use `FSNewAliasMinimal` (page 16) instead.)

`NewAliasMinimalFromFullPath` (page 42) Deprecated in Mac OS X v10.4
Creates an alias record that contains only the full pathname of the target. (Deprecated. Use `FSNewAliasMinimal` (page 16) or `FSNewAliasMinimalUnicode` (page 17) instead.)

`ResolveAlias` (page 43) Deprecated in Mac OS X v10.4
Identifies the single most likely target of an alias record. (Deprecated. Use `FSResolveAlias` (page 19) instead.)

`ResolveAliasFile` (page 44) Deprecated in Mac OS X v10.4
Resolves an alias contained in an alias file. (Deprecated. Use `FSResolveAliasFile` (page 20) instead.)

`ResolveAliasFileWithMountFlagsNoUI` (page 45) Deprecated in Mac OS X v10.4
Resolves an alias file without any user interaction. (Deprecated. Use `FSResolveAliasFileWithMountFlags` (page 21) with the `kResolveAliasFileNoUI` flag instead.)

`ResolveAliasWithMountFlags` (page 46) Deprecated in Mac OS X v10.4
Identifies the target of an alias. (Deprecated. Use `FSResolveAliasWithMountFlags` (page 22) instead.)

`UpdateAlias` (page 47) Deprecated in Mac OS X v10.4
Updates an alias record. (Deprecated. Use `FSUpdateAlias` (page 23) instead.)

Functions

DisposeAliasFilterUPP

Disposes of a universal procedure pointer (UPP) to an alias filtering callback function.

```
void DisposeAliasFilterUPP (
   AliasFilterUPP userUPP
);
```

Parameters

`userUPP`
The UPP to dispose of.

Discussion

See `AliasFilterProcPtr` (page 28) for more information on alias filtering callback functions.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Aliases.h`

FSCopyAliasInfo

Returns information from an alias handle.

```
OSStatus FSCopyAliasInfo (
   AliasHandle inAlias,
   HFSUniStr255 *targetName,
   HFSUniStr255 *volumeName,
   CFStringRef *pathString,
   FSAliasInfoBitmap *whichInfo,
   FSAliasInfo *info
);
```

Parameters

inAlias

A handle to the alias record from which to get information.

targetName

A pointer to a string that, on return, contains the name of the target item. Pass `NULL` if you do not want this information returned.

volumeName

A pointer to a string that, on return, contains the name of the volume the target resides on. Pass `NULL` if you do not want this information returned.

pathString

A pointer a CFString that, on return, contains the POSIX path to the target. Pass `NULL` if you do not want this information returned.

whichInfo

A pointer to a variable of type `FSAliasInfoBitmap`. On return, this field indicates which fields in the alias information block, specified in the `info` parameter, contain valid data. See "Alias Information Masks" (page 32) for a description of the values that may be returned here. This parameter may be `NULL`.

info

A pointer to a structure of type `FSAliasInfo` (page 30). On return, this structure contains information about the alias. Pass `NULL` if you do not want this information returned.

Return Value

A result code.

Discussion

This function returns the requested information from the alias handle passed in the *inAlias* parameter. The information is gathered only from the alias record, so it may not match what is on disk. No disk input/output is performed.

The `FSCopyAliasInfo` function adds support for unicode filenames and filenames longer than 32 bytes. It replaces the `GetAliasInfo` function.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Aliases.h`

FSFollowFinderAlias

Resolves an alias record obtained from a Finder alias file.

```
OSErr FSFollowFinderAlias (
   FSRef *fromFile,
   AliasHandle alias,
   Boolean logon,
   FSRef *target,
   Boolean *wasChanged
);
```

Parameters

fromFile
: A pointer to the file to use for a first attempt at a relative resolution; pass a pointer to the alias file's `FSRef` for this parameter.

alias
: A handle to the alias record taken from the alias file's resources.

logon
: If `true`, the Alias Manager attempts to mount a volume if necessary to complete the resolution of the alias.

target
: A pointer to an `FSRef` structure. On return, this `FSRef` refers to the target found by the resolution.

wasChanged
: A pointer to a Boolean value. `FSFollowFinderAlias` sets this value to `true` if it has updated the alias record.

Return Value
A result code.

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

Declared In
`Aliases.h`

FSIsAliasFile

Determines whether a file system object is an alias file, a data file, or a folder.

```
OSErr FSIsAliasFile (
   const FSRef *fileRef,
   Boolean *aliasFileFlag,
   Boolean *folderFlag
);
```

Parameters

fileRef
: A pointer to the file system object to test.

aliasFileFlag
: A pointer to a Boolean variable. On return, a value of `TRUE` indicates that the object specified in the *fileRef* parameter is an alias file. A value of `FALSE` indicates that the object is not an alias file.

folderFlag

A pointer to a Boolean variable. On return, a value of `TRUE` indicates that the object specified in the *fileRef* parameter is a folder. A value of `FALSE` indicates that the object is a file.

Return Value

A result code.

Discussion

Table 1 summarizes the information that this function provides about the object specified in the `fileRef` parameter:

Table 1 Information about a file system object

Alias flag	Folder flag	Object kind
T	F	Alias file
F	F	Data file
F	T	Folder

Note that if `fileRef` is an alias file, this function does not provide any information about the object to which the alias refers. To find out whether this object is a file or a folder, you can use `FSResolveAliasFile` (page 20).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTCarbonShell

Declared In

`Aliases.h`

FSMatchAliasBulk

Identifies a list of possible matches for an alias.

```
OSStatus FSMatchAliasBulk (
    const FSRef *fromFile,
    unsigned long rulesMask,
    AliasHandle inAlias,
    short *aliasCount,
    FSRef *aliasList,
    Boolean *needsUpdate,
    FSAliasFilterProcPtr aliasFilter,
    void *yourDataPtr
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. You may pass `NULL` if you do not want this function to perform a relative search. By default, this function performs a relative search only if the absolute search does not find a match. If you want to perform the relative search first, you should pass `kARMSearchRelFirst` in the `rulesMask` parameter.

rulesMask

A set of rules to guide the resolution. Pass the sum of all of the rules you want to invoke. For a description of the values you can use in this parameter, see "Matching Constants" (page 33).

inAlias

A handle to the alias record to be resolved.

aliasCount

On input, a pointer to the maximum number of possible matches to return. On output, the actual number of matches returned.

aliasList

A pointer to an array of `FSRef` structures. On output, this array holds the results of the search, a list of possible candidates.

needsUpdate

A pointer to a Boolean flag that, on output, indicates whether the alias record needs to be updated. For more information about this parameter, see the Discussion.

aliasFilter

An optional application-defined filter function. The Alias Manager calls your filter function each time it identifies a possible match or after the search has continued for three seconds without a match. Your filter function returns a Boolean value that determines whether the possible match is discarded (`true`) or added to the list of possible targets (`false`). It can also terminate the search by setting the variable parameter `quitFlag`. See `FSAliasFilterProcPtr` (page 28) for a description of the filter function.

yourDataPtr

A pointer to data to be passed to the filter function, or `NULL`. The `yourDataPtr` parameter can point to any data your application might need in the filter function.

Return Value

A result code. If the Alias Manager finds the specified volume and parent directory but fails to find the target file or directory in that location, the return value is `fnfErr` and the elements in the `aliasList` parameter are not valid.

Discussion

After it identifies a target, this function compares some key information about the target with the same information in the record. If the information does not match, this function sets the *`needsUpdate`* flag to `true`. This function also sets the *`needsUpdate`* flag to `true` if it identifies a list of possible matches rather than a single match or if `kARMsearchRelFirst` is set in the *`rulesMask`* parameter but the target is identified through either an absolute search or an exhaustive search. Otherwise, this function sets the *needsUpdate* flag to `false`. This function always sets the *needsUpdate* flag to `false` when resolving an alias created by `FSNewAliasMinimal`. To update the alias record to reflect the final results of the resolution, use the function `FSUpdateAlias` (page 23).

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

Declared In
`Aliases.h`

FSNewAlias

Creates a new alias record, given a target file or directory.

```
OSErr FSNewAlias (
   const FSRef *fromFile,
   const FSRef *target,
   AliasHandle *inAlias
);
```

Parameters

fromFile
: A pointer to the starting point for a relative search. You may pass `NULL` if you do not need relative search information in the alias record. The files or directories specified in the *fromFile* and *target* parameters must reside on the same volume.

target
: A pointer to the target file or directory of the alias.

inAlias
: A pointer to an alias handle. On return, this handle refers to the newly created alias record. If the function fails to create an alias record, it sets *inAlias* to `NULL`.

Return Value
A result code. If the specified target is valid, this function creates an alias record for the target and returns `noErr`. Any other return value indicates that this function did not create an alias record.

Discussion
The `FSNewAlias` function creates an alias record that describes the specified target. It allocates the storage, fills in the record, and puts a record handle to that storage in the *inAlias* parameter. `FSNewAlias` records the full pathname of the target and a collection of other information relevant to locating the target, verifying the target, and mounting the target's volume, if necessary. You can have `FSNewAlias` store relative search information as well by supplying a starting point for a relative search.

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

Related Sample Code
QTCarbonShell

Declared In
`Aliases.h`

FSNewAliasFromPath

Creates a new alias record, given the pathname of the target file or directory.

```
OSErr FSNewAliasFromPath (
   const char *fromFilePath,
   const char *targetPath,
   OptionBits flags,
   AliasHandle *inAlias,
   Boolean *isDirectory
);
```

Parameters

fromFilePath

A C string that specifies the starting point for a relative search. The string should contain a UTF-8 pathname. You may pass `NULL` if you do not need relative search information in the alias record.

targetPath

A C string that contains the full UTF-8 pathname of the target object.

flags

Reserved for future use. Currently, you should pass 0.

inAlias

A pointer to an alias handle. On output, this handle refers to the newly created alias record.

isDirectory

A pointer to a Boolean value. On input, if the target does not exist, set the value to `true` if the target is a directory or `false` if it is not. (Pass `NULL` if you are not sure whether the target is a directory.) On output, if the target exists, the value is `true` if the target is a directory, `false` if it is not.

Return Value

A result code. For more information, see the Discussion.

Discussion

If the specified target exists, this function creates an alias record for the target and returns `noErr`. If the parent directory specified in the target pathname exists but the target itself does not exist, this function creates an alias record for the target and returns `fnfErr`. Any other return value indicates that this function did not create an alias record.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`Aliases.h`

FSNewAliasMinimal

Creates a new minimal alias record, given a target file or directory.

```
OSErr FSNewAliasMinimal (
   const FSRef *target,
   AliasHandle *inAlias
);
```

Parameters

target

A pointer to the target of the alias record.

inAlias

A pointer to an alias handle. On return, this handle refers to the newly created alias record. If the function fails to create an alias record, it sets `inAlias` to `NULL`.

Return Value

A result code. If the specified target is valid, this function creates an alias record for the target and returns `noErr`. Any other return value indicates that this function did not create an alias record.

Discussion

The `FSNewAliasMinimal` function creates an alias record that contains only the minimum information necessary to describe the target. The `FSNewAliasMinimal` function uses the standard alias record data structure, but it fills in only parts of the record.

The `FSResolveAlias` (page 19) function never updates a minimal alias record.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Aliases.h`

FSNewAliasMinimalUnicode

Creates a minimal alias, given the Unicode name and parent directory of the target.

```
OSErr FSNewAliasMinimalUnicode (
   const FSRef *targetParentRef,
   UniCharCount targetNameLength,
   const UniChar *targetName,
   AliasHandle *inAlias,
   Boolean *isDirectory
);
```

Parameters

targetParentRef

A pointer to the parent directory of the target.

targetNameLength

The number of Unicode characters in the target's name.

targetName

A pointer to the Unicode name of the target.

inAlias

A pointer to an alias handle. On return, this handle refers to the newly created alias record.

isDirectory

A pointer to a Boolean value. On input, if the target does not exist, set the value to `true` if the target is a directory or `false` if it is not. (Pass `NULL` if you are not sure whether the target is a directory.) On output, if the target exists, the value is `true` if the target is a directory, `false` if it is not.

Return Value
A result code. For more information, see the Discussion.

Discussion
If the specified target exists, this function creates an alias record for the target and returns `noErr`. If the parent directory exists but the target itself does not exist, this function creates an alias record for the target and returns `fnfErr`. Any other return value indicates that this function did not create an alias record.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Aliases.h`

FSNewAliasUnicode

Creates a new alias record, given the Unicode name and parent directory of the target.

```
OSErr FSNewAliasUnicode (
    const FSRef *fromFile,
    const FSRef *targetParentRef,
    UniCharCount targetNameLength,
    const UniChar *targetName,
    AliasHandle *inAlias,
    Boolean *isDirectory
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. You may pass `NULL` if you do not need relative search information in the alias record.

targetParentRef

A pointer to the parent directory of the target.

targetNameLength

The number of Unicode characters in the target's name.

targetName

A pointer to the Unicode name of the target.

inAlias

A pointer to an alias handle. On return, this handler refers to the newly created alias record.

isDirectory

A pointer to a Boolean value. On input, if the target does not exist, set the value to `true` if the target is a directory or `false` if it is not. (Pass `NULL` if you are not sure whether the target is a directory.) On output, if the target exists, the value is `true` if the target is a directory, `false` if it is not.

Return Value
A result code. For more information, see the Discussion.

Discussion

If the specified target exists, this function creates an alias record for the target and returns `noErr`. If the parent directory exists but the target itself does not exist, this function creates an alias record for the target and returns `fnfErr`. Any other return value indicates that this function did not create an alias record.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Aliases.h`

FSResolveAlias

Returns an `FSRef` to the single most likely target of an alias record.

```
OSErr FSResolveAlias (
   const FSRef *fromFile,
   AliasHandle alias,
   FSRef *target,
   Boolean *wasChanged
);
```

Parameters

`fromFile`

A pointer to the starting point for a relative search. If you pass `NULL` in this parameter, `FSResolveAlias` performs only an absolute search. If you pass a pointer to a valid `FSRef` in the `fromFile` parameter, `FSResolveAlias` performs a relative search for the target, followed by an absolute search only if the relative search fails. If you want to perform an absolute search followed by a relative search, you should use the function `FSMatchAliasBulk` (page 13).

`alias`

A handle to the alias record to be resolved and, if necessary, updated.

`target`

A pointer to an `FSRef`. On successful return, this `FSRef` describes the target of the alias record. This parameter must point to a valid `FSRef` structure.

`wasChanged`

A pointer to a Boolean value indicating, on return, whether the alias record in the *`alias`* parameter was updated because it contained some outdated information about the target. If it updates the alias record, `FSResolveAlias` sets the `wasChanged` parameter to `true`. Otherwise, it sets it to `false`. (`FSResolveAlias` never updates a minimal alias, so it never sets `wasChanged` to `true` when resolving a minimal alias.

Return Value

A result code. When it finds the specified volume and parent directory but fails to find the target file or directory in that location, `FSResolveAlias` returns `fnfErr`. Note that the `FSRef` in the `alias` parameter is not valid in this case.

Discussion

The `FSResolveAlias` function performs a fast search for the target of the alias. If the resolution is successful, `FSResolveAlias` returns (in the *`target`* parameter) the `FSRef` for the target file system object, updates the alias record if necessary, and reports (through the *`wasChanged`* parameter) whether the record was updated. If the target is on an unmounted AppleShare volume, `FSResolveAlias` automatically mounts the volume. If the target is on an unmounted ejectable volume, `FSResolveAlias` asks the user to insert the volume. The `FSResolveAlias` function exits after it finds one acceptable target.

After it identifies a target, `FSResolveAlias` compares some key information about the target with the information in the alias record. If the information differs, `FSResolveAlias` updates the record to match the target.

The `FSResolveAlias` function displays the standard dialogs when it needs input from the user, such as a name and password for mounting a remote volume. The user can cancel the resolution through these dialogs.

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

Related Sample Code
QTCarbonShell

Declared In
`Aliases.h`

FSResolveAliasFile

Resolves an alias contained in an alias file.

```
OSErr FSResolveAliasFile (
   FSRef *theRef,
   Boolean resolveAliasChains,
   Boolean *targetIsFolder,
   Boolean *wasAliased
);
```

Parameters

theRef
: A pointer to the alias file you plan to open. If the function completes successfully, this `FSRef` describes to the file or the directory referred to by the alias file.

resolveAliasChains
: A Boolean value. Set this parameter to `TRUE` if you want `FSResolveAliasFile` to resolve all aliases in a chain (for example, an alias file that refers to an alias file and so on), stopping only when it reaches the target file. Set this parameter to `FALSE` if you want to resolve only one alias file, even if the target is another alias file.

targetIsFolder
: A pointer to a Boolean value. The `FSResolveAliasFile` function returns `TRUE` in this parameter if the `FSRef` in the parameter `theRef` points to a directory or a volume; otherwise, `FSResolveAliasFile` returns `FALSE` in this parameter.

wasAliased
: A pointer to a Boolean value. The `FSResolveAliasFile` function returns `TRUE` in this parameter if the `FSRef` in the parameter `theRef` points to an alias; otherwise, `FSResolveAliasFile` returns `FALSE` in this parameter.

Return Value
A result code. When it finds the specified volume and parent directory but fails to find the target file or directory in that location, `FSResolveAliasFile` returns `fnfErr`.

Discussion
If your application bypasses the Finder when manipulating documents, it should check for and resolve aliases itself by using the `FSResolveAliasFile` function.

The `FSResolveAliasFile` function first checks the catalog file for the file or directory specified in the parameter *`theRef`* to determine whether it is an alias and whether it is a file or a directory. If the object is not an alias, `FSResolveAliasFile` leaves *`theRef`* unchanged, sets the *`targetIsFolder`* parameter to `TRUE` for a directory or volume and `FALSE` for a file, sets *`wasAliased`* to `FALSE`, and returns `noErr`. If the object is an alias, `FSResolveAliasFile` resolves it, places the target in the parameter *`theRef`*, and sets the *`wasAliased`* flag to `TRUE`.

If `FSResolveAliasFile` receives an error code while resolving an alias, it leaves the input parameters as they are and exits, returning an error code. `FSResolveAliasFile` can return any Resource Manager or File Manager errors.

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

Declared In
`Aliases.h`

FSResolveAliasFileWithMountFlags

Resolves an alias contained in an alias file.

```
OSErr FSResolveAliasFileWithMountFlags (
   FSRef *theRef,
   Boolean resolveAliasChains,
   Boolean *targetIsFolder,
   Boolean *wasAliased,
   unsigned long mountFlags
);
```

Parameters

`theRef`
: A pointer to the alias file you plan to open. If the function completes successfully, this `FSRef` describes the file or the directory referred to by the alias file.

`resolveAliasChains`
: A Boolean value. Set this parameter to `TRUE` if you want `FSResolveAliasFileWithMountFlags` to resolve all aliases in a chain (for example, an alias file that refers to an alias file and so on), stopping only when it reaches the target file. Set this parameter to `FALSE` if you want to resolve only one alias file, even if the target is another alias file.

`targetIsFolder`
: A pointer to a Boolean value. The `FSResolveAliasFileWithMountFlags` function returns `TRUE` in this parameter if the `FSRef` in the parameter `theRef` points to a directory or a volume; otherwise, `FSResolveAliasFileWithMountFlags` returns `FALSE` in this parameter.

`wasAliased`
: A pointer to a Boolean value. The `FSResolveAliasFileWithMountFlags` function returns `TRUE` in this parameter if the `FSRef` in the parameter `theRef` points to an alias; otherwise, `FSResolveAliasFileWithMountFlags` returns `FALSE` in this parameter.

`mountFlags`
: Options controlling how the alias file is resolved. See "Volume Mount Options" (page 33) for a description of the values you can use here. Set this parameter to `kResolveAliasFileNoUI` to prevent any user interaction, including disk switch alerts, while the alias is being resolved.

Return Value
A result code.

Discussion
The function `FSResolveAliasFileWithMountFlags` is identical to `FSResolveAliasFile` (page 20) with the exception that it provides the *`mountFlags`* parameter, allowing callers additional control over how the alias file is resolved.

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

Declared In
`Aliases.h`

FSResolveAliasWithMountFlags

Returns an `FSRef` to the target of an alias.

```
OSErr FSResolveAliasWithMountFlags (
   const FSRef *fromFile,
   AliasHandle inAlias,
   FSRef *target,
   Boolean *wasChanged,
   unsigned long mountFlags
);
```

Parameters

`fromFile`

A pointer to the starting point for a relative search. If you pass `NULL` in this parameter, `FSResolveAliasWithMountFlags` performs an absolute search. If you pass a pointer to a valid `FSRef` in the `fromFile` parameter, `FSResolveAliasWithMountFlags` performs a relative search for the target, followed by an absolute search only if the relative search fails. If you want to perform an absolute search followed by a relative search, you should use the function `FSMatchAliasBulk` (page 13).

`inAlias`

A handle to the alias record to be resolved and, if necessary, updated.

`target`

A pointer to an `FSRef` structure. On successful return, this `FSRef` refers to the target of the alias record. This parameter must point to a valid `FSRef` structure.

`wasChanged`

A pointer to a Boolean value indicating, on return, whether the alias record to be resolved was updated because it contained some outdated information about the target. If it updates the alias record, `FSResolveAliasWithMountFlags` sets the `wasChanged` parameter to `true`. Otherwise, it sets it to `false`. (`FSResolveAliasWithMountFlags` never updates a minimal alias, so it never sets `wasChanged` to `true` when resolving a minimal alias.

`mountFlags`

Options controlling how the alias is resolved. See "Volume Mount Options" (page 33) for a description of the values you can use here. Set this parameter to `kResolveAliasFileNoUI` to prevent any user interaction while the alias is being resolved.

Return Value
A result code.

Discussion
The function `FSResolveAliasWithMountFlags` is identical to `FSResolveAlias` (page 19) with the exception that it provides the `mountFlags` parameter, allowing callers additional control over how the alias is resolved.

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

Declared In
`Aliases.h`

FSUpdateAlias

Updates an alias record for a specified target.

```
OSErr FSUpdateAlias (
   const FSRef *fromFile,
   const FSRef *target,
   AliasHandle alias,
   Boolean *wasChanged
);
```

Parameters

`fromFile`
: A pointer to the starting point for a relative search. You may pass `NULL` if you do not need relative search information in the alias record. The two files or directories specified in the `fromFile` and `target` parameters must reside on the same volume.

`target`
: A pointer to the target of the alias record.

`alias`
: A handle to the alias record to be updated.

`wasChanged`
: A pointer to a Boolean value that, on output, indicates whether the newly constructed alias record is different from the old one. If the new record is exactly the same as the old one, the value is `false`. Otherwise, the value is `true`. Check this parameter to determine whether you need to save an updated record.

Return Value
A result code.

Discussion
This function rebuilds the entire alias record and fills it in as the `FSNewAlias` function would. The `FSUpdateAlias` function always creates a complete alias record. When you use `FSUpdateAlias` to update a minimal alias record, you convert the minimal record to a complete record.

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

Declared In
`Aliases.h`

GetAliasSize

Gets the size of an alias record referenced by a handle.

```
Size GetAliasSize (
   AliasHandle alias
);
```

Parameters

alias

A handle to the alias record from which to get the information.

Return Value

The size of the alias record.

Discussion

The returned size is smaller than the size returned by the function `GetHandleSize` if any custom data is added. This routine is thread safe.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Aliases.h`

GetAliasSizeFromPtr

Gets the size of an alias record referenced by a pointer.

```
Size GetAliasSizeFromPtr (
   const AliasRecord *alias
);
```

Parameters

alias

A pointer to the alias record from which to get the information.

Return Value

The size of the alias record.

Discussion

This routine is thread safe.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Aliases.h`

GetAliasUserType

Gets the user type for an alias record referenced by a handle.

```
OSType GetAliasUserType (
   AliasHandle alias
);
```

Parameters

alias

A handle to the alias record from which to get the user type.

Return Value

The user type associated with the alias.

Discussion

This routine is thread safe.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Aliases.h`

GetAliasUserTypeFromPtr

Gets the user type for the alias record referenced by a pointer.

```
OSType GetAliasUserTypeFromPtr (
   const AliasRecord *alias
);
```

Parameters

alias

A pointer to the alias record from which to get the user type.

Return Value

The user type associated with the alias.

Discussion

This routine is thread safe.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Aliases.h`

InvokeAliasFilterUPP

Calls your alias filtering callback function.

```
Boolean InvokeAliasFilterUPP (
   CInfoPBPtr cpbPtr,
   Boolean *quitFlag,
   Ptr myDataPtr,
   AliasFilterUPP userUPP
);
```

Discussion
You should not need to use the function `InvokeAliasFilterUPP`, as the system calls your alias filtering callback for you.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Aliases.h`

NewAliasFilterUPP

Creates a new universal procedure pointer (UPP) to an alias filtering callback function.

```
AliasFilterUPP NewAliasFilterUPP (
   AliasFilterProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your alias filtering callback function. For more information, see `AliasFilterProcPtr` (page 28).

Return Value
On return, a UPP to the alias filtering callback function.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Aliases.h`

SetAliasUserType

Sets the user type for an alias record referenced by a handle.

```
void SetAliasUserType (
   AliasHandle alias,
   OSType userType
);
```

Parameters

alias

A handle to the alias record for which to set the user type.

userType

The user type associated with the alias.

Discussion

This routine is thread safe.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Aliases.h`

SetAliasUserTypeWithPtr

Sets the user type for the alias record referenced by a pointer.

```
void SetAliasUserTypeWithPtr (
   AliasPtr alias,
   OSType userType
);
```

Parameters

alias

A pointer to the alias record for which to set the user type.

userType

The user type associated with the alias.

Discussion

This routine is thread safe.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Aliases.h`

Callbacks

AliasFilterProcPtr

Defines a pointer to an alias filtering callback function that filters out possible targets identified by the `FSMatchAlias` (page 49) function.

```
typedef Boolean (*AliasFilterProcPtr) (
    CInfoPBPtr cpbPtr,
    Boolean * quitFlag,
    Ptr myDataPtr
);
```

If you name your function `MyAliasFilterCallback`, you would declare it like this:

```
Boolean MyAliasFilterCallback (
    CInfoPBPtr cpbPtr,
    Boolean * quitFlag,
    Ptr myDataPtr
);
```

Parameters

cpbPtr

A pointer to a catalog information parameter block. When your function is called, the `cpbPtr` parameter points to the catalog information parameter block of the possible match (returned by the File Manager function `PBGetCatInfo`).

quitFlag

On exit, set this to `true` if you want to terminate the search.

myDataPtr

A pointer to any customized data that your application passed when it called `FSMatchAlias` (page 49). This parameter allows your filter function to access any data that your application has set up on its own.

Return Value

Your function should return `true` to indicate that the possible match is to be discarded, or `false` to indicate that the possible match is to be added to the list of possible targets.

Discussion

You can write your own filter function to examine possible targets identified by the `FSMatchAlias` function. The `FSMatchAlias` function calls your filter function each time it identifies a possible match.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`Aliases.h`

FSAliasFilterProcPtr

Defines a pointer to an alias filtering callback function that filters out possible targets identified by the `FSMatchAliasBulk` (page 13) function.

```
typedef Boolean (*FSAliasFilterProcPtr) (
    FSRef *ref,
    Boolean *quitFlag,
    Ptr myDataPtr
);
```

If you name your function `MyFSAliasFilterCallback`, you would declare it like this:

```
Boolean MyAliasFilterCallback (
    FSRef *ref,
    Boolean *quitFlag,
    Ptr myDataPtr
);
```

Parameters

`ref`

A pointer to a file system object. When your function is called, the *`ref`* parameter points to the possible match.

`quitFlag`

On output, set this Boolean flag to `true` if you want to terminate the search.

`myDataPtr`

A pointer to any customized data that your application passed when it called `FSMatchAliasBulk` (page 13). This parameter allows your filter function to access any data that your application has set up on its own.

Return Value

Your function should return `true` to indicate that the possible match is to be discarded, or `false` to indicate that the possible match is to be added to the list of possible targets.

Discussion

You can write your own filter function to examine possible targets identified by the `FSMatchAliasBulk` function. The `FSMatchAliasBulk` function calls your filter function each time it identifies a possible match.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`Aliases.h`

Data Types

AliasInfoType

Defines the alias record information type used in the index parameter of `GetAliasInfo`.

```
typedef short AliasInfoType;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Aliases.h`

AliasFilterUPP

Defines a universal procedure pointer (UPP) to an alias filtering function.

```
typedef AliasFilterProcPtr AliasFilterUPP;
```

Discussion

See `AliasFilterProcPtr` (page 28) for more information on alias filtering functions.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`Aliases.h`

AliasRecord

Defines an alias record.

```
struct AliasRecord {
    OSType userType;
    unsigned short aliasSize;
};
typedef struct AliasRecord            AliasRecord;
typedef AliasRecord *                 AliasPtr;
typedef AliasPtr *                    AliasHandle;
```

Fields

`userType`
: A 4-byte field that can contain application-specific data. When an alias record is created, this field contains 0. Your application can use this field for its own purposes.

`aliasSize`
: The size, in bytes, assigned to the alias record at the time of its creation or updating. This is the total size of the record, including the `userType` and `aliasSize` fields, as well as the variable-length data that is private to the Alias Manager.

Discussion
The Alias Manager uses alias records to store information that allows it to locate an object in the file system.

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

Declared In
`Aliases.h`

FSAliasInfo

Defines an information block passed to the `FSCopyAliasInfo` function.

```
struct FSAliasInfo {
    UTCDateTime volumeCreateDate;
    UTCDateTime targetCreateDate;
    OSType fileType;
    OSType fileCreator;
    UInt32 parentDirID;
    UInt32 nodeID;
    UInt16 filesystemID;
    UInt16 signature;
    Boolean volumeIsBootVolume;
    Boolean volumeIsAutomounted;
    Boolean volumeIsEjectable;
    Boolean volumeHasPersistentFileIDs;
    Boolean isDirectory;
};
typedef struct FSAliasInfo FSAliasInfo;
typedef FSAliasInfo * FSAliasInfoPtr;
```

Fields

`volumeCreateDate`
: The creation date of the volume on which the alias target resides.

`targetCreateDate`
: The creation date of the alias target.

`fileType`
: The file type of the target.

`fileCreator`
: The creator code of the target.

`parentDirID`
: The directory ID of the target's parent directory.

`nodeID`
: The ID of the file or directory that is the alias target.

`filesystemID`
: The filesystem ID.

`signature`
: The volume signature of the volume on which the target resides.

`volumeIsBootVolume`
: A Boolean value indicating whether the volume is the boot volume.

`volumeIsAutomounted`
: A Boolean value indicating whether the volume is automounted.

`volumeIsEjectable`
: A Boolean value indicating whether the volume is ejectable.

`volumeHasPersistentFileIDs`
: A Boolean value indicating whether the volume has persistent file ID's.

`isDirectory`
: A Boolean value indicating whether the alias target is a directory.

Availability

Available in Mac OS X v10.2 and later.

Declared In
`Aliases.h`

Constants

Alias Information Masks

Returned by the `FSCopyAliasInfo` function to indicate which fields of the alias information structure contain valid data.

```
typedef UInt32 FSAliasInfoBitmap;
enum {
    kFSAliasInfoNone = 0x00000000,
    kFSAliasInfoVolumeCreateDate = 0x00000001,
    kFSAliasInfoTargetCreateDate = 0x00000002,
    kFSAliasInfoFinderInfo = 0x00000004,
    kFSAliasInfoIsDirectory = 0x00000008,
    kFSAliasInfoIDs = 0x00000010,
    kFSAliasInfoFSInfo = 0x00000020,
    kFSAliasInfoVolumeFlags = 0x00000040
};
```

Constants

`kFSAliasInfoNone`

None of the alias information is valid.

Available in Mac OS X v10.2 and later.

Declared in `Aliases.h`.

`kFSAliasInfoVolumeCreateDate`

The volume creation date in the `volumeCreateDate` field is valid.

Available in Mac OS X v10.2 and later.

Declared in `Aliases.h`.

`kFSAliasInfoTargetCreateDate`

The creation date of the alias target, in the `targetCreateDate` field, is valid.

Available in Mac OS X v10.2 and later.

Declared in `Aliases.h`.

`kFSAliasInfoFinderInfo`

The file type and creator information, in the `fileType` and `fileCreator` fields, is valid.

Available in Mac OS X v10.2 and later.

Declared in `Aliases.h`.

`kFSAliasInfoIsDirectory`

The information in the `isDirectory` field is valid.

Available in Mac OS X v10.2 and later.

Declared in `Aliases.h`.

`kFSAliasInfoIDs`
: The parent directory ID and alias target ID, in the `parentDirID` and `nodeID` fields, are valid.

  Available in Mac OS X v10.2 and later.

  Declared in `Aliases.h`.

`kFSAliasInfoFSInfo`
: The filesystem ID and signature, in the `filesystemID` and `signature` fields, are valid.

  Available in Mac OS X v10.2 and later.

  Declared in `Aliases.h`.

`kFSAliasInfoVolumeFlags`
: The volume information, in the `volumeIsBootVolume`, `volumeIsAutomounted`, `volumeIsEjectable`, and `volumeHasPersistentFileIDs` fields, is valid.

  Available in Mac OS X v10.2 and later.

  Declared in `Aliases.h`.

Volume Mount Options

Specify how an alias should be resolved.

```
enum {
    kResolveAliasFileNoUI = 0x00000001,
    kResolveAliasTryFileIDFirst = 0x00000002
};
```

Constants

`kResolveAliasFileNoUI`
: The Alias Manager should resolve the alias without presenting a user interface.

  Available in Mac OS X v10.0 and later.

  Declared in `Aliases.h`.

`kResolveAliasTryFileIDFirst`
: The Alias Manager should search for the alias target using file IDs before searching using the path.

  Available in Mac OS X v10.2 and later.

  Declared in `Aliases.h`.

Discussion

The `FSResolveAliasWithMountFlags` (page 22), `FSResolveAliasFileWithMountFlags` (page 21), `ResolveAliasWithMountFlags` (page 46), `ResolveAliasFileWithMountFlags` (page 53), and `ResolveAliasFileWithMountFlagsNoUI` (page 45) functions take these constants in the `mountFlags` parameter, allowing you to specify how the alias should be resolved.

Matching Constants

Specify the matching criteria for the alias matching functions.

```
enum {
    kARMMountVol = 0x00000001,
    kARMNoUI = 0x00000002,
    kARMMultVols = 0x00000008,
    kARMSearch = 0x00000100,
    kARMSearchMore = 0x00000200,
    kARMSearchRelFirst = 0x00000400,
    kARMTryFileIDFirst = 0x00000800
};
```

Constants

`kARMMountVol`

Automatically try to mount the target's volume if it is not mounted.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`kARMNoUI`

Stop if a search requires user interaction, such as a password dialog box when mounting a remote volume. If user interaction is needed and `kARMNoUI` is in effect, the search fails.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`kARMMultVols`

Search all mounted volumes. The search begins with the volume on which the target resided when the record was created. When you specify a fast search of all mounted volumes, `MatchAlias` performs a formal fast search only on the volume described in the alias record. On all other volumes it looks for the target by ID or by name in the directory with the specified parent directory ID. When you specify an exhaustive search of multiple volumes, `MatchAlias` performs the same search on all volumes. When resolving an alias record created by `NewAliasMinimalFromFullPath`, `MatchAlias` ignores this flag.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`kARMSearch`

Perform a fast search for the alias target. If `kARMSearchRelFirst` is not set, perform an absolute search first, followed by a relative search only if the value of the `fromFile` parameter is not `NULL` and the list of matches is not full.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`kARMSearchMore`

Perform an exhaustive search for the alias target. On HFS volumes, the exhaustive search uses the File Manager function `PBCatSearch` to identify candidates with matching creation date, type, and creator. The `PBCatSearch` function is available only on HFS volumes and only on systems running version 7.0 or later. On MFS volumes or HFS volumes that do not support `PBCatSearch`, the exhaustive search makes a series of indexed calls to File Manager functions, using the same search criteria. If you set `kARMSearchMore` and either or both of `kARMSearch` and `kARMSearchRelFirst`, `MatchAlias` performs the fast search first.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`kARMSearchRelFirst`
: If `kARMSearch` is also set, perform a relative search before the absolute search. (If `kARMSearch` is also set and the target is found through the absolute search, `MatchAlias` sets the `needsUpdate` flag to `true`.) If neither `kARMSearch` nor `kARMSearchMore` is set, perform only a relative search. If `kARMSearch` is not set but `kARMSearchMore` is set, perform a relative search followed by an exhaustive search.

  Available in Mac OS X v10.0 and later.

  Declared in `Aliases.h`.

`kARMTryFileIDFirst`
: Perform a search using the file ID of the target before searching using the path.

  Available in Mac OS X v10.2 and later.

  Declared in `Aliases.h`.

Discussion

The `FSMatchAlias` (page 49), `FSMatchAliasNoUI` (page 50), `MatchAliasNoUI` (page 52) and `MatchAlias` (page 39) functions use these constants to specify the matching criteria by passing a sum of these constants in the *`rulesMask`* parameter. You must specify at least one of the last three parameters: `kARMSearch`, `kARMSearchMore`, and `kARMSearchRelFirst`.

Alias Resource Type

Specifies the file type of an alias resource file.

```
enum {
    rAliasType = 'alis'
};
```

Information Type Constants

The `GetAliasInfo` function uses these constants in the `index` parameter.

```
enum {
    asiZoneName = -3,
    asiServerName = -2,
    asiVolumeName = -1,
    asiAliasName = 0,
    asiParentName = 1
};
```

Constants

`asiZoneName`
: If the record represents a target on an AppleShare volume, retrieve the server's zone name. Otherwise, return an empty string.

  Available in Mac OS X v10.0 and later.

  Declared in `Aliases.h`.

`asiServerName`
: If the record represents a target on an AppleShare volume, retrieve the server name. Otherwise, return an empty string.

  Available in Mac OS X v10.0 and later.

  Declared in `Aliases.h`.

`asiVolumeName`
Return the name of the volume on which the target resides.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`asiAliasName`
Return the name of the target.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

`asiParentName`
Return the name of the parent directory of the target of the record. If the target is a volume, return the volume name.

Available in Mac OS X v10.0 and later.

Declared in `Aliases.h`.

Gestalt Constants

You can check for version and feature availability information by using the Alias Manager selectors defined in the Gestalt Manager. For more information, see *Gestalt Manager Reference*.

Deprecated Alias Manager Functions

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

Deprecated in Mac OS X v10.3

GetAliasInfo

Gets information from an alias record without actually resolving the record. (Deprecated in Mac OS X v10.3. Use `FSCopyAliasInfo` (page 11) instead.)

```
OSErr GetAliasInfo (
   AliasHandle alias,
   AliasInfoType index,
   Str63 theString
);
```

Parameters

`alias`

A handle to the alias record to be read.

`index`

The kind of information to be retrieved. If the value of `index` is a positive integer, `GetAliasInfo` retrieves the parent directory that has the same hierarchical level above the target as the index parameter (for example, an `index` value of 2 returns the name of the parent directory of the target's parent directory). You can therefore assemble the names of the target and all of its parent directories by making repeated calls to `GetAliasInfo` with incrementing index values, starting with a value of 0. When the value of `index` is greater than the number of levels between the target and the root, `GetAliasInfo` returns an empty string. You can also set the `index` parameter to one of the values described in "Information Type Constants" (page 35).

`theString`

A string that, on return, holds the requested information.

Return Value

A result code.

Discussion

The `GetAliasInfo` function returns the information stored in the alias record, which might not be current. To ensure that the information is current, you can resolve and update the alias record before calling `GetAliasInfo`.

The `GetAliasInfo` function cannot provide all kinds of information about a minimal alias.

Special Considerations

Use the `FSCopyAliasInfo` (page 11) function instead of `GetAliasInfo`. `GetAliasInfo` does not reliably return information for aliases to items on POSIX file systems. In addition, `GetAliasInfo` does not support unicode names or names longer than 32 bytes. If the name of the alias target is longer than 32 bytes, the name is truncated and the file ID and extension (if any) are appended before the name is returned by `GetAliasInfo`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`Aliases.h`

Deprecated in Mac OS X v10.4

IsAliasFile

Determines whether a file system object is an alias file, a data file, or a folder. (Deprecated in Mac OS X v10.4. Use `FSIsAliasFile` (page 12) instead.)

```
OSErr IsAliasFile (
   const FSSpec *fileFSSpec,
   Boolean *aliasFileFlag,
   Boolean *folderFlag
);
```

Parameters

fileFSSpec

A pointer to a file specification structure describing a file.

aliasFileFlag

A pointer to a Boolean variable. On return, a value of `TRUE` indicates that the object specified in the *fileRef* parameter is an alias file. A value of `FALSE` indicates that the object is not an alias file.

folderFlag

A pointer to a Boolean variable. On return, a value of `TRUE` indicates that the object specified in the *fileRef* parameter is a folder. A value of `FALSE` indicates that the object is a file.

Return Value

A result code.

Discussion

This function determines whether a file is an alias file.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`Aliases.h`

MatchAlias

Identifies a list of possible matches for an alias and passes the list through an optional selection filter. The filter can return more than one possible match. (Deprecated in Mac OS X v10.4. Use `FSMatchAliasBulk` (page 13) instead.)

```
OSErr MatchAlias (
   const FSSpec *fromFile,
   unsigned long rulesMask,
   AliasHandle alias,
   short *aliasCount,
   FSSpecArrayPtr aliasList,
   Boolean *needsUpdate,
   AliasFilterUPP aliasFilter,
   void *yourDataPtr
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. If you do not want `MatchAlias` to perform a relative search, set `fromFile` to `NULL`. If you want `MatchAlias` to perform a relative search, pass a pointer to a file system specification structure that describes the starting point for the search.

rulesMask

A set of rules to guide the resolution. Pass the sum of all of the rules you want to invoke. For a description of the values you can use in this parameter, see "Matching Constants" (page 33).

alias

A handle to the alias record to be resolved.

aliasCount

On input, a pointer to the maximum number of possible matches to return. On output, the actual number of matches returned.

aliasList

A pointer to the array that holds the results of the search, a list of possible candidates.

needsUpdate

A pointer to a Boolean flag that indicates whether the alias record to be resolved needs to be updated.

aliasFilter

An application-defined filter function. The Alias Manager executes this function each time it identifies a possible match and after the search has continued for three seconds without a match. Your filter function returns a Boolean value that determines whether the possible match is discarded (`true`) or added to the list of possible targets (`false`). It can also terminate the search by setting the variable parameter `quitFlag`. See `AliasFilterProcPtr` (page 28) for a description of the filter function.

yourDataPtr

A pointer to data to be passed to the filter function. The `yourDataPtr` parameter can point to any data your application might need in the filter function.

Return Value

A result code.

Discussion

If `MatchAlias` finds the parent directory on the correct volume but does not find the target, it sets the *aliasCount* parameter to 1, puts the file system specification structure for the target in the results list, and returns `fnfErr`. The `FSSpec` structure is valid, although the object it describes does not exist. This information is intended as a "hint" that lets you explore possible solutions to the resolution failure. You can, for example, use the `FSSpec` structure and the File Manager function `FSpCreate` to create a replacement for a missing file.

After it identifies a target, `MatchAlias` compares some key information about the target with the same information in the record. If the information does not match, `MatchAlias` sets the *`needsUpdate`* flag to `true`. The key information is

- the name of the target
- the directory ID of the target's parent
- the file ID or directory ID of the target
- the name and creation date of the volume on which the target resides

The `MatchAlias` function also sets the *`needsUpdate`* flag to `true` if it identifies a list of possible matches rather than a single match or if `kARMsearchRelFirst` is set in the `rulesMask` parameter but the target is identified through either an absolute search or an exhaustive search. Otherwise, the `MatchAlias` function sets the *`needsUpdate`* flag to `false`. `MatchAlias` always sets the *`needsUpdate`* flag to `false` when resolving an alias created by `NewAliasMinimal`. If you want to update the alias record to reflect the final results of the resolution, call `UpdateAlias`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`Aliases.h`

NewAlias

Creates a complete alias record. (Deprecated in Mac OS X v10.4. Use `FSNewAlias` (page 15) instead.)

```
OSErr NewAlias (
   const FSSpec *fromFile,
   const FSSpec *target,
   AliasHandle *alias
);
```

Parameters

`fromFile`
A pointer to the starting point for a relative search. If you do not need relative search information in the alias record, pass a `fromFile` value of `NULL`. If you want `NewAlias` to record relative search information, pass a pointer to a valid `FSSpec` structure in this parameter. The files or directories specified in the `fromFile` and `target` parameters must reside on the same volume.

`target`
A pointer to an `FSSpec` structure for the target of the alias record.

`alias`
A pointer to an alias handle. On return, this handle refers to the newly created alias record. If the function fails to create an alias record, it sets *`alias`* to `NULL`.

Return Value
A result code.

Discussion
The `NewAlias` function creates an alias record that describes the specified target. It allocates the storage, fills in the record, and puts a record handle to that storage in the *`alias`* parameter. `NewAlias` always records the name and file or directory ID of the target, its creation date, the parent directory name and ID, and the volume name and creation date. It also records the full pathname of the target and a collection of other information relevant to locating the target, verifying the target, and mounting the target's volume, if necessary. You can have `NewAlias` store relative search information as well by supplying a starting point for a relative search.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`Aliases.h`

NewAliasMinimal

Creates a short alias record quickly. (Deprecated in Mac OS X v10.4. Use `FSNewAliasMinimal` (page 16) instead.)

```
OSErr NewAliasMinimal (
   const FSSpec *target,
   AliasHandle *alias
);
```

Parameters

`target`
A pointer to the target of the alias record.

`alias`
A pointer to an alias handle. On return, this handle refers to the newly created alias record. If the function fails to create an alias record, it sets `alias` to `NULL`.

Return Value
A result code.

Discussion
The `NewAliasMinimal` function creates an alias record that contains only the minimum information necessary to describe the target: the target name, the parent directory ID, the volume name and creation date, and the volume mounting information. The `NewAliasMinimal` function uses the standard alias record data structure, but it fills in only parts of the record.

The `ResolveAlias` (page 43) function never updates a minimal alias record.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`Aliases.h`

NewAliasMinimalFromFullPath

Creates an alias record that contains only the full pathname of the target. (Deprecated in Mac OS X v10.4. Use `FSNewAliasMinimal` (page 16) or `FSNewAliasMinimalUnicode` (page 17) instead.)

```
OSErr NewAliasMinimalFromFullPath (
   short fullPathLength,
   const void *fullPath,
   ConstStr32Param zoneName,
   ConstStr31Param serverName,
   AliasHandle *alias
);
```

Parameters

`fullPathLength`
: The number of characters in the full pathname of the target.

`fullPath`
: A pointer to a buffer that contains the full pathname of the target. The full pathname starts with the name of the volume, includes all of the directory names in the path to the target, and ends with the target name. (For a description of pathnames, see the documentation for the File Manager.)

`zoneName`
: The AppleTalk zone name of the AppleShare volume on which the target resides. Set this parameter to a null string if you do not need it.

`serverName`
: The AppleTalk server name of the AppleShare volume on which the target resides. Set this parameter to a null string if you do not need it.

`alias`
: A pointer to an alias handle. On return, this handle refers to the newly created alias record. If the function fails to create an alias record, it sets `alias` to `NULL`.

Return Value
A result code.

Discussion
The `NewAliasMinimalFromFullPath` function creates an alias record that identifies the target by full pathname. You can call `NewAliasMinimalFromFullPath` to create an alias record for a file that doesn't exist or that resides on an unmounted volume.

The `NewAliasMinimalFromFullPath` function uses the standard alias record data structure, but it fills in only the information provided in the input parameters. You can therefore use `NewAliasMinimalFromFullPath` to create alias records for targets on unmounted volumes.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`Aliases.h`

ResolveAlias

Identifies the single most likely target of an alias record. (Deprecated in Mac OS X v10.4. Use `FSResolveAlias` (page 19) instead.)

```
OSErr ResolveAlias (
   const FSSpec *fromFile,
   AliasHandle alias,
   FSSpec *target,
   Boolean *wasChanged
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. If you pass a `fromFile` parameter of `NULL`, `ResolveAlias` performs only an absolute search. If you pass a pointer to a valid `FSSpec` structure in the `fromFile` parameter, `ResolveAlias` performs a relative search for the target, followed by an absolute search only if the relative search fails. If you want to perform an absolute search followed by a relative search, you must use the `MatchAlias` function.

alias

A handle to the alias record to be resolved and, if necessary, updated.

target

A pointer to the target of the alias record. This parameter must be a valid `FSSpec` structure.

wasChanged

A pointer to a Boolean value indicating whether the alias record to be resolved was updated because it contained some outdated information about the target. If it updates the alias record, `ResolveAlias` sets the `wasChanged` parameter to `true`. Otherwise, it sets it to `false`. (`ResolveAlias` never updates a minimal alias, so it never sets `wasChanged` to `true` when resolving a minimal alias.

Return Value

A result code.

Discussion

The `ResolveAlias` function performs a fast search for the target of the alias. If the resolution is successful, `ResolveAlias` returns (in the *`target`* parameter) the `FSSpec` structure for the target file system object, updates the alias record if necessary, and reports (through the *`wasChanged`* parameter) whether the record was updated. If the target is on an unmounted AppleShare volume, `ResolveAlias` automatically mounts the volume. If the target is on an unmounted ejectable volume, `ResolveAlias` asks the user to insert the volume. The `ResolveAlias` function exits after it finds one acceptable target.

After it identifies a target, `ResolveAlias` compares some key information about the target with the information in the alias record. (The description of the `MatchAlias` (page 39) function lists the key information.) If the information differs, `ResolveAlias` updates the record to match the target.

When it finds the specified volume and parent directory but fails to find the target file or directory in that location, `ResolveAlias` returns a result code of `fnfErr` and fills in the *`target`* parameter with a complete `FSSpec` structure describing the target (that is, the volume reference number, parent directory ID, and filename or folder name). The `FSSpec` structure is valid, although the object it describes does not exist. This information is intended as a "hint" that lets you explore possible solutions to the resolution failure. You can, for example, pass the `FSSpec` structure to the File Manager function `FSpCreate` to create a replacement for a missing file.

The `ResolveAlias` function displays the standard dialog boxes when it needs input from the user, such as a name and password for mounting a remote volume. The user can cancel the resolution through these dialog boxes.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`Aliases.h`

ResolveAliasFile

Resolves an alias contained in an alias file. (Deprecated in Mac OS X v10.4. Use `FSResolveAliasFile` (page 20) instead.)

```
OSErr ResolveAliasFile (
   FSSpec *theSpec,
   Boolean resolveAliasChains,
   Boolean *targetIsFolder,
   Boolean *wasAliased
);
```

Parameters

`theSpec`

A pointer to the alias file you plan to open. If the function completes successfully, this `FSSpec` refers to the file or the directory that was referred to by the alias file.

`resolveAliasChains`

A Boolean value. Set this parameter to `TRUE` if you want `ResolveAliasFile` to resolve all aliases in a chain (for example, an alias file that refers to an alias file and so on), stopping only when it reaches the target file. Set this parameter to `FALSE` if you want to resolve only one alias file, even if the target is another alias file.

`targetIsFolder`

A return parameter only. The `ResolveAliasFile` function returns `TRUE` in this parameter if the file specification structure in the parameter `theSpec` points to a directory or a volume; otherwise, `ResolveAliasFile` returns `FALSE` in this parameter.

`wasAliased`

A return parameter only. The `ResolveAliasFile` function returns `TRUE` in this parameter if the file specification structure in the parameter `theSpec` points to an alias; otherwise, `ResolveAliasFile` returns `FALSE` in this parameter.

Return Value
A result code.

Discussion
If your application bypasses the Finder when manipulating documents, it should check for and resolve aliases itself by using the `ResolveAliasFile` function.

The `ResolveAliasFile` function first checks the catalog file for the file or directory specified in the parameter *`theSpec`* to determine whether it is an alias and whether it is a file or a directory. If the object is not an alias, `ResolveAliasFile` leaves *`theSpec`* unchanged, sets the *`targetIsFolder`* parameter to `TRUE` for a directory or volume and `FALSE` for a file, sets *`wasAliased`* to `FALSE`, and returns `noErr`. If the object is an alias, `ResolveAliasFile` resolves it, places the target in the parameter *`theSpec`*, and sets the *`wasAliased`* flag to `TRUE`.

When `ResolveAliasFile` finds the specified volume and parent directory but fails to find the target file or directory in that location, `ResolveAliasFile` returns a result code of `fnfErr` and fills in the parameter *`theSpec`* with a complete file system specification structure describing the target (that is, its volume reference number, parent directory ID, and filename or folder name). The file system specification structure is valid, although the object it describes does not exist. This information is intended as a "hint" that lets you explore possible solutions to the resolution failure. You can, for example, use the file system specification structure to create a replacement for a missing file with the File Manager function `FSpCreate`.

If `ResolveAliasFile` receives an error code while resolving an alias, it leaves the input parameters as they are and exits, returning an error code. `ResolveAliasFile` can return any Resource Manager or File Manager errors.

Special Considerations

Before calling the `ResolveAliasFile` function, you should make sure that it is available by using the `Gestalt` function with the `gestaltAliasMgrAttr` selector.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`Aliases.h`

ResolveAliasFileWithMountFlagsNoUI

Resolves an alias file without any user interaction. (Deprecated in Mac OS X v10.4. Use `FSResolveAliasFileWithMountFlags` (page 21) with the `kResolveAliasFileNoUI` flag instead.)

```
OSErr ResolveAliasFileWithMountFlagsNoUI (
   FSSpec *theSpec,
   Boolean resolveAliasChains,
   Boolean *targetIsFolder,
   Boolean *wasAliased,
   unsigned long mountFlags
);
```

Parameters

`theSpec`
: A pointer to the alias file you plan to open. If the function completes successfully, this `FSSpec` refers to the file or the directory that was referred to by the alias file.

`resolveAliasChains`
: A Boolean value. Set this parameter to `TRUE` if you want `ResolveAliasFileWithMountFlagsNoUI` to resolve all aliases in a chain (for example, an alias file that refers to an alias file and so on), stopping only when it reaches the target file. Set this parameter to `FALSE` if you want to resolve only one alias file, even if the target is another alias file.

targetIsFolder

A return parameter only. The `ResolveAliasFileWithMountFlagsNoUI` function returns `TRUE` in this parameter if the file specification structure in the parameter `theSpec` points to a directory or a volume; otherwise, `ResolveAliasFileWithMountFlagsNoUI` returns `FALSE` in this parameter.

wasAliased

A return parameter only. The `ResolveAliasFileWithMountFlagsNoUI` function returns `TRUE` in this parameter if the file specification structure in the parameter `theSpec` points to an alias; otherwise, `ResolveAliasFileWithMountFlagsNoUI` returns `FALSE` in this parameter.

mountFlags

Options controlling how the alias file is resolved. See "Volume Mount Options" (page 33) for a description of the values you can use here. Set this parameter to `kResolveAliasFileNoUI` to prevent any user interaction, including disk switch alerts, while the alias is being resolved.

Return Value
A result code.

Discussion
The function `ResolveAliasFileWithMountFlagsNoUI` is identical to `ResolveAliasFile` (page 44) with the exception that it presents no interface to the user.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`Aliases.h`

ResolveAliasWithMountFlags

Identifies the target of an alias. (Deprecated in Mac OS X v10.4. Use `FSResolveAliasWithMountFlags` (page 22) instead.)

```
OSErr ResolveAliasWithMountFlags (
   const FSSpec *fromFile,
   AliasHandle alias,
   FSSpec *target,
   Boolean *wasChanged,
   unsigned long mountFlags
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. If you pass `NULL` in this parameter, `ResolveAliasWithMountFlags` performs only an absolute search. If you pass a pointer to a valid `FSSpec` structure in the `fromFile` parameter, `ResolveAliasWithMountFlags` performs a relative search for the target, followed by an absolute search only if the relative search fails. If you want to perform an absolute search followed by a relative search, you must use the `MatchAlias` function.

alias

A handle to the alias record to be resolved and, if necessary, updated.

target

A pointer to an `FSSpec` structure. On return, this `FSSpec` identifies the target of the alias record. This parameter must point to a valid `FSSpec` structure.

wasChanged

A pointer to a Boolean value indicating, on return, whether the alias record to be resolved was updated because it contained some outdated information about the target. If it updates the alias record, `ResolveAliasWithMountFlags` sets the `wasChanged` parameter to `true`. Otherwise, it sets it to `false`. (`ResolveAliasWithMountFlags` never updates a minimal alias, so it never sets `wasChanged` to `true` when resolving a minimal alias.

mountFlags

Options controlling how the alias is resolved. See "Volume Mount Options" (page 33) for a description of the values you can use here. Set this parameter to `kResolveAliasFileNoUI` to prevent any user interaction while the alias is being resolved.

Return Value

A result code.

Discussion

The function `ResolveAliasWithMountFlags` is identical to `ResolveAlias` (page 43) with the exception that it provides the `mountFlags` parameter, allowing callers additional control over how the alias is resolved.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`Aliases.h`

UpdateAlias

Updates an alias record. (Deprecated in Mac OS X v10.4. Use `FSUpdateAlias` (page 23) instead.)

```
OSErr UpdateAlias (
   const FSSpec *fromFile,
   const FSSpec *target,
   AliasHandle alias,
   Boolean *wasChanged
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. If you do not need relative search information in the record, pass a `fromFile` value of `NULL`. If you want `UpdateAlias` to record relative search information, pass a pointer to a valid `FSSpec` structure in this parameter.

target

A pointer to the target of the alias record.

alias

A handle to the alias record to be updated.

wasChanged
A pointer to a Boolean value indicating whether the newly constructed alias record is exactly the same as the old one. If the new record is the same as the old one, `UpdateAlias` sets the `wasChanged` parameter to `false`. Otherwise, it sets it to `true`. Check this parameter to determine whether you need to save an updated record.

Return Value
A result code.

Discussion
The `UpdateAlias` function rebuilds the entire alias record and fills it in as the `NewAlias` function would.

The `UpdateAlias` function always creates a complete alias record. When you use `UpdateAlias` to update a minimal alias record, you convert the minimal record to a complete record.

Special Considerations

The two files or directories, specified in the `fromFile` and `target` parameters, must reside on the same volume.

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

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`Aliases.h`

Deprecated in Mac OS X v10.5

FollowFinderAlias

Resolves an alias record obtained from a Finder alias file. (Deprecated in Mac OS X v10.5. Use `FSFollowFinderAlias` (page 12) instead.)

```
OSErr FollowFinderAlias (
   const FSSpec *fromFile,
   AliasHandle alias,
   Boolean logon,
   FSSpec *target,
   Boolean *wasChanged
);
```

Parameters

fromFile
A pointer to a file system specification specifying a file for a first attempt at a relative resolution; pass a pointer to the alias file's `FSSpec` forthis parameter.

alias
A handle to the alias record taken from the alias file's resources.

logon
If `true`, the Alias Manager attempts to mount a volume if necessary to complete the resolution of the alias.

target

A pointer to an `FSSpec` structure. On return, this `FSSpec` refers to the target found by the resolution.

wasChanged

A pointer to a Boolean value. `FollowFinderAlias` sets this value to `true` if it has updated the alias record. If the alias has been updated, you should call `ChangedResource` and `WriteResource` if the updated record should be saved in the resource file.

Return Value

A result code.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Aliases.h`

FSMatchAlias

Identifies a list of possible matches for an alias. (Deprecated in Mac OS X v10.5. Use `FSMatchAliasBulk` (page 13) instead.)

```
OSErr FSMatchAlias (
   const FSRef *fromFile,
   unsigned long rulesMask,
   AliasHandle inAlias,
   short *aliasCount,
   FSRef *aliasList,
   Boolean *needsUpdate,
   AliasFilterUPP aliasFilter,
   void *yourDataPtr
);
```

Parameters

fromFile

A pointer to the starting point for a relative search. You may pass `NULL` if you do not want this function to perform a relative search.

rulesMask

A set of rules to guide the resolution. Pass the sum of all of the rules you want to invoke. For a description of the values you can use in this parameter, see "Matching Constants" (page 33).

inAlias

A handle to the alias record to be resolved.

aliasCount

On input, a pointer to the maximum number of possible matches to return. On output, the actual number of matches returned.

aliasList

A pointer to an array of `FSRef` structures. On return, this array holds the results of the search, a list of possible candidates.

needsUpdate

A pointer to a Boolean flag that, on return, indicates whether the alias record needs to be updated.

aliasFilter
: An application-defined filter function. The Alias Manager executes this function each time it identifies a possible match. Your filter function returns a Boolean value that determines whether the possible match is discarded (`true`) or added to the list of possible targets (`false`). It can also terminate the search by setting the variable parameter `quitFlag`. See `AliasFilterProcPtr` (page 28) for a description of the filter function.

yourDataPtr
: A pointer to data to be passed to the filter function. The `yourDataPtr` parameter can point to any data your application might need in the filter function.

Return Value
A result code. When it finds the specified volume and parent directory but fails to find the target file or directory in that location, `FSMatchAlias` returns `fnfErr`. Note that the file system objects in the `aliasList` parameter are not valid in this case.

Discussion
After it identifies a target, `FSMatchAlias` compares some key information about the target with the same information in the record. If the information does not match, `FSMatchAlias` sets the *needsUpdate* flag to `true`.

The `FSMatchAlias` function also sets the *needsUpdate* flag to `true` if it identifies a list of possible matches rather than a single match or if `kARMsearchRelFirst` is set in the *rulesMask* parameter but the target is identified through either an absolute search or an exhaustive search. Otherwise, the `FSMatchAlias` function sets the *needsUpdate* flag to `false`. `FSMatchAlias` always sets the *needsUpdate* flag to `false` when resolving an alias created by `FSNewAliasMinimal`. If you want to update the alias record to reflect the final results of the resolution, call `FSUpdateAlias`.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`Aliases.h`

FSMatchAliasNoUI

Identifies a list of possible matches for an alias without any user interaction. (Deprecated in Mac OS X v10.5. Use `FSMatchAliasBulk` (page 13) with the `kARMNoUI` flag instead.)

```
OSErr FSMatchAliasNoUI (
   const FSRef *fromFile,
   unsigned long rulesMask,
   AliasHandle inAlias,
   short *aliasCount,
   FSRef *aliasList,
   Boolean *needsUpdate,
   AliasFilterUPP aliasFilter,
   void *yourDataPtr
);
```

Parameters

fromFile
: A pointer to the starting point for a relative search. You may pass `NULL` if you do not want this function to perform a relative search.

rulesMask
: A set of rules to guide the resolution. Pass the sum of all of the rules you want to invoke. For a description of the values you can use in this parameter, see "Matching Constants" (page 33).

inAlias
: A handle to the alias record to be resolved.

aliasCount
: On input, a pointer to the maximum number of possible matches to return. On output, the actual number of matches returned.

aliasList
: A pointer to an array of `FSRef` structures. On return, this array holds the results of the search, a list of possible candidates.

needsUpdate
: A pointer to a Boolean flag that, on return, indicates whether the alias record needs to be updated.

aliasFilter
: An application-defined filter function. The Alias Manager executes this function each time it identifies a possible match. Your filter function returns a Boolean value that determines whether the possible match is discarded (`true`) or added to the list of possible targets (`false`). It can also terminate the search by setting the variable parameter `quitFlag`. See `AliasFilterProcPtr` (page 28) for a description of the filter function.

yourDataPtr
: A pointer to data to be passed to the filter function. The `yourDataPtr` parameter can point to any data your application might need in the filter function.

Return Value
A result code.

Discussion
The `FSMatchAliasNoUI` function operates in much the same way as the `FSMatchAlias` function; however, it does not present an interface to the user. Additionally, the `FSMatchAliasNoUI` function does not mount network volumes, even when it is possible to mount the volume without user interaction. See the discussion of `FSMatchAlias` (page 49) for more information.

Availability
Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`Aliases.h`

MatchAliasNoUI

Identifies a list of possible matches for an alias without any user interaction. (Deprecated in Mac OS X v10.5. Use `FSMatchAliasBulk` (page 13) with the `kARMNoUI` flag instead.)

```
OSErr MatchAliasNoUI (
   const FSSpec *fromFile,
   unsigned long rulesMask,
   AliasHandle alias,
   short *aliasCount,
   FSSpecArrayPtr aliasList,
   Boolean *needsUpdate,
   AliasFilterUPP aliasFilter,
   void *yourDataPtr
);
```

Parameters

`fromFile`
: A pointer to the starting point for a relative search. If you do not want `MatchAliasNoUI` to perform a relative search, set `fromFile` to `NULL`. If you want `MatchAliasNoUI` to perform a relative search, pass a pointer to a file system specification structure that describes the starting point for the search.

`rulesMask`
: A set of rules to guide the resolution. Pass the sum of all of the rules you want to invoke. For a description of the values you can use in this parameter, see "Matching Constants" (page 33).

`alias`
: A handle to the alias record to be resolved.

`aliasCount`
: On input, a pointer to the maximum number of possible matches to return. On output, the actual number of matches returned.

`aliasList`
: A pointer to the array of `FSSpec` structures that holds, on return, the results of the search, a list of possible candidates.

`needsUpdate`
: A pointer to a Boolean flag that, on return, indicates whether the alias record needs to be updated.

`aliasFilter`
: An application-defined filter function. The Alias Manager executes this function each time it identifies a possible match and after the search has continued for three seconds without a match. Your filter function returns a Boolean value that determines whether the possible match is discarded (`true`) or added to the list of possible targets (`false`). It can also terminate the search by setting the variable parameter `quitFlag`. See `AliasFilterProcPtr` (page 28) for a description of the filter function.

`yourDataPtr`
: A pointer to data to be passed to the filter function. The `yourDataPtr` parameter can point to any data your application might need in the filter function.

Return Value
A result code.

Discussion
The MatchAliasNoUI function operates in the same way as the MatchAlias function; however, it does not present an interface to the user. See the discussion of MatchAlias (page 39) for more information.

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

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
Aliases.h

ResolveAliasFileWithMountFlags

Resolves an alias contained in an alias file. (Deprecated in Mac OS X v10.5. Use FSResolveAliasFileWithMountFlags (page 21) instead.)

```
OSErr ResolveAliasFileWithMountFlags (
   FSSpec *theSpec,
   Boolean resolveAliasChains,
   Boolean *targetIsFolder,
   Boolean *wasAliased,
   unsigned long mountFlags
);
```

Parameters

theSpec
A pointer to the alias file you plan to open. If the function completes successfully, this FSSpec refers to the file or the directory that was referred to by the alias file.

resolveAliasChains
A Boolean value. Set this parameter to TRUE if you want ResolveAliasFileWithMountFlags to resolve all aliases in a chain (for example, an alias file that refers to an alias file and so on), stopping only when it reaches the target file. Set this parameter to FALSE if you want to resolve only one alias file, even if the target is another alias file.

targetIsFolder
A return parameter only. The ResolveAliasFileWithMountFlags function returns TRUE in this parameter if the file specification structure in the parameter theSpec points to a directory or a volume; otherwise, ResolveAliasFileWithMountFlags returns FALSE in this parameter.

wasAliased
A return parameter only. The ResolveAliasFileWithMountFlags function returns TRUE in this parameter if the file specification structure in the parameter theSpec points to an alias; otherwise, ResolveAliasFileWithMountFlags returns FALSE in this parameter.

mountFlags
Options controlling how the alias file is resolved. See "Volume Mount Options" (page 33) for a description of the values you can use here. Set this parameter to kResolveAliasFileNoUI to prevent any user interaction while the alias is being resolved.

Return Value
A result code.

Discussion

The function `ResolveAliasFileWithMountFlags` is identical to `ResolveAliasFile` (page 44) with the exception that it provides the *`mountFlags`* parameter.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`Aliases.h`

Document Revision History

This table describes the changes to *Alias Manager Reference*.

Date	Notes
2006-12-05	Updated for Mac OS X v10.5.
2006-07-24	Updated to describe replacements for deprecated functions.
2005-11-09	Fixed typographical errors in several function names.
2005-08-11	Updated documentation of functions FSIsAliasFile and IsAliasFile.
2005-04-29	Updated for Mac OS X v10.4.
2003-01-02	Added documentation for the Alias Manager functions that use the `FSRef` type.

Index