CFBundle Reference

Core Foundation

2007-10-18

Contents

CFBundle Reference

Derived From:	*CFType Reference*
Framework:	CoreFoundation/CoreFoundation.h
Declared in	CFBundle.h
Companion guides	Bundle Programming Guide Plug-ins

Overview

CFBundle allows you to use a folder hierarchy called a bundle to organize and locate many types of application resources including images, sounds, localized strings, and executable code. In Mac OS X, bundles can also be used by CFM applications to load and execute functions from Mach-O frameworks. You can use bundles to support multiple languages or execute your application on multiple operating environments.

You create a bundle object using one of the `CFBundleCreate...` functions. CFBundle provides several functions for finding resources within a bundle. The `CFBundleCopyResourceURL` (page 19) function returns the location of a resource of the specified name and type, and in the specified subdirectory. Use `CFBundleCopyResourceURLForLocalization` (page 19) to restrict the search to a specific localization name. Use `CFBundleCopyResourceURLsOfType` (page 21) to get the locations of all resources of a specified type.

CFBundle provides functions for getting bundle information, such as its identifier and information dictionary. Use the `CFBundleGetIdentifier` (page 30) function to get the identifier of a bundle, and the `CFBundleGetInfoDictionary` (page 31) function to get its information dictionary. The principal intended purpose for locating bundles by identifier is so that code (in frameworks, plugins, etc.) can find its own bundle.

You can also obtain locations of subdirectories in a bundle represented as CFURL objects. The `CFBundleCopyExecutableURL` (page 13) function returns the location of the application's executable. The functions `CFBundleCopyResourceURL` (page 19), `CFBundleCopySharedFrameworksURL` (page 23), `CFBundleCopyPrivateFrameworksURL` (page 17), `CFBundleCopySharedSupportURL` (page 24), and `CFBundleCopyBuiltInPlugInsURL` (page 11) return the location of a bundle's subdirectory containing resources, shared frameworks, private frameworks, shared support files, and plug-ins respectively.

Other functions are used to manage localizations. The `CFBundleCopyLocalizedString` (page 16) and `CFBundleCopyLocalizationsForURL` (page 16) functions return a localized string from a bundle's strings file. The `CFBundleCopyLocalizationsForPreferences` (page 15) function returns the localizations that CFBundle would prefer, given the specified bundle and user preference localizations.

Unlike some other Core Foundation opaque types with similar Cocoa Foundation names (such as CFString and `NSString`), `NSBundle` objects cannot be cast ("toll-free bridged") to CFBundle objects.

Unlike NSBundle, which does not support unloading (because the Objective C runtime does not support the unloading of Objective C code), you can unload CFBundle objects.

`CFBundleGetFunctionPointerForName` (page 29) and related calls automatically load a bundle if it is not already loaded. When the last reference to the CFBundle object is released and it is finally deallocated, then the code will be unloaded if it is still loaded and if the executable is of a type that supports unloading. If you keep this in mind, and if you make sure that everything that uses the bundle keeps a retain on the CFBundle object, then you can just use the bundle naturally and never have to worry about when it is loaded and unloaded.

On the other hand, if you want to manually manage when the bundle is loaded and unloaded, then you can use `CFBundleLoadExecutable` (page 36) and `CFBundleUnloadExecutable` (page 39)—although this technique is not recommended. These functions force immediate loading and unloading of the executable (if it has not already been loaded/unloaded, and in the case of unloading if the executable is of a type that supports unloading). If you do this, then the code calling `CFBundleUnloadExecutable` is responsible for making sure that there are no remaining references to anything in the bundle's code before it is unloaded. In the previous approach, by contrast, this responsibility can be distributed to the individual code sections that use the bundle, by making sure that each one keeps its own retain on the CFBundle object.

One further point about CFBundle reference counting: if you are taking the first approach, but do not actually wish the bundle's code to be unloaded (as is often the case), or if you are taking the second approach of manually managing the unloading yourself, then in many cases you do not actually have to worry about releasing a CFBundle object. CFBundle instances are uniqued, so there is only one CFBundle object for a given bundle, and rarely are there so many bundles being considered at once that the memory usage for CFBundle objects would be significant. There are cases in which a process could create CFBundle objects for potentially an unlimited number of bundles, and such processes would wish to balance retains and releases carefully, but such cases are likely to be rare.

Note that it is best to compile any unloadable bundles with the flag `-fno-constant-cfstrings`—see *Bundle Programming Guide* for more details.

Functions by Task

Creating and Accessing Bundles

`CFBundleCreate` (page 25)
    Creates a CFBundle object.

`CFBundleCreateBundlesFromDirectory` (page 26)
    Searches a directory and constructs an array of CFBundle objects from all valid bundles in the specified directory.

`CFBundleGetAllBundles` (page 26)
    Returns an array containing all of the bundles currently open in the application.

`CFBundleGetBundleWithIdentifier` (page 27)
    Locate a bundle given its program-defined identifier.

`CFBundleGetMainBundle` (page 32)
    Returns an application's main bundle.

Loading and Unloading a Bundle

`CFBundleIsExecutableLoaded` (page 35)
: Obtains information about the load status for a bundle's main executable.

`CFBundlePreflightExecutable` (page 38)
: Returns a Boolean value that indicates whether a given bundle is loaded or appears to be loadable.

`CFBundleLoadExecutable` (page 36)
: Loads a bundle's main executable code into memory and dynamically links it into the running application.

`CFBundleLoadExecutableAndReturnError` (page 36)
: Returns a Boolean value that indicates whether a given bundle is loaded, attempting to load it if necessary.

`CFBundleUnloadExecutable` (page 39)
: Unloads the main executable for the specified bundle.

Finding Locations in a Bundle

`CFBundleCopyAuxiliaryExecutableURL` (page 10)
: Returns the location of a bundle's auxiliary executable code.

`CFBundleCopyBuiltInPlugInsURL` (page 11)
: Returns the location of a bundle's built in plug-in.

`CFBundleCopyExecutableURL` (page 13)
: Returns the location of a bundle's main executable code.

`CFBundleCopyPrivateFrameworksURL` (page 17)
: Returns the location of a bundle's private Frameworks directory.

`CFBundleCopyResourcesDirectoryURL` (page 18)
: Returns the location of a bundle's Resources directory.

`CFBundleCopySharedFrameworksURL` (page 23)
: Returns the location of a bundle's shared frameworks directory.

`CFBundleCopySharedSupportURL` (page 24)
: Returns the location of a bundle's shared support files directory.

`CFBundleCopySupportFilesDirectoryURL` (page 24)
: Returns the location of the bundle's support files directory.

Locating Bundle Resources

`CFBundleCloseBundleResourceMap` (page 10)
: Closes an open resource map for a bundle.

`CFBundleCopyResourceURL` (page 19)
: Returns the location of a resource contained in the specified bundle.

`CFBundleCopyResourceURLInDirectory` (page 20)
: Returns the location of a resource contained in the specified bundle directory without requiring the creation of a CFBundle object.

`CFBundleCopyResourceURLsOfType` (page 21)
: Assembles an array of URLs specifying all of the resources of the specified type found in a bundle.

`CFBundleCopyResourceURLsOfTypeInDirectory` (page 23)
: Returns an array of CFURL objects describing the locations of all resources in a bundle of the specified type without needing to create a CFBundle object.

`CFBundleCopyResourceURLForLocalization` (page 19)
: Returns the location of a localized resource in a bundle.

`CFBundleCopyResourceURLsOfTypeForLocalization` (page 22)
: Returns an array containing copies of the URL locations for a specified bundle, resource, and localization name.

`CFBundleOpenBundleResourceFiles` (page 37)
: Opens the non-localized and localized resource files (if any) for a bundle in separate resource maps.

`CFBundleOpenBundleResourceMap` (page 38)
: Opens the non-localized and localized resource files (if any) for a bundle in a single resource map.

Managing Localizations

`CFBundleCopyBundleLocalizations` (page 11)
: Returns an array containing a bundle's localizations.

`CFBundleCopyLocalizedString` (page 16)
: Returns a localized string from a bundle's strings file.

`CFBundleCopyLocalizationsForPreferences` (page 15)
: Given an array of possible localizations and preferred locations, returns the one or more of them that CFBundle would use, without reference to the current application context.

`CFBundleCopyLocalizationsForURL` (page 16)
: Returns an array containing the localizations for a bundle or executable at a particular location.

`CFBundleCopyPreferredLocalizationsFromArray` (page 17)
: Given an array of possible localizations, returns the one or more of them that CFBundle would use in the current application context.

`CFCopyLocalizedString` (page 39)
: Searches the default strings file `Localizable.strings` for the string associated with the specified key.

`CFCopyLocalizedStringFromTable` (page 40)
: Searches the specified strings file for the string associated with the specified key.

`CFCopyLocalizedStringFromTableInBundle` (page 40)
: Returns a localized version of the specified string.

`CFCopyLocalizedStringWithDefaultValue` (page 41)
: Returns a localized version of a localization string.

Managing Executable Code

`CFBundleGetDataPointerForName` (page 27)
: Returns a data pointer to a symbol of the given name.

`CFBundleGetDataPointersForNames` (page 28)
: Returns a C array of data pointer to symbols of the given names.

`CFBundleGetFunctionPointerForName` (page 29)
: Returns a pointer to a function in a bundle's executable code using the function name as the search key.

`CFBundleGetFunctionPointersForNames` (page 30)
: Constructs a function table containing pointers to all of the functions found in a bundle's main executable code.

`CFBundleGetPlugIn` (page 33)
: Returns a bundle's plug-in.

Getting Bundle Properties

`CFBundleCopyBundleURL` (page 12)
: Returns the location of a bundle.

`CFBundleGetDevelopmentRegion` (page 28)
: Returns the bundle's development region from the bundle's information property list.

`CFBundleGetIdentifier` (page 30)
: Returns the bundle identifier from a bundle's information property list.

`CFBundleGetInfoDictionary` (page 31)
: Returns a bundle's information dictionary.

`CFBundleGetLocalInfoDictionary` (page 31)
: Returns a bundle's localized information dictionary.

`CFBundleGetValueForInfoDictionaryKey` (page 34)
: Returns a value (localized if possible) from a bundle's information dictionary.

`CFBundleCopyInfoDictionaryInDirectory` (page 14)
: Returns a bundle's information dictionary.

`CFBundleCopyInfoDictionaryForURL` (page 14)
: Returns the information dictionary for a given URL location.

`CFBundleGetPackageInfo` (page 32)
: Returns a bundle's package type and creator.

`CFBundleGetPackageInfoInDirectory` (page 33)
: Returns a bundle's package type and creator without having to create a CFBundle object.

`CFBundleCopyExecutableArchitectures` (page 12)
: Returns an array of CFNumbers representing the architectures a given bundle provides.

`CFBundleCopyExecutableArchitecturesForURL` (page 13)
: Returns an array of CFNumbers representing the architectures a given URL provides.

`CFBundleGetVersionNumber` (page 35)
: Returns a bundle's version number.

Getting the CFBundle Type ID

`CFBundleGetTypeID` (page 34)
: Returns the type identifier for the CFBundle opaque type.

Functions

CFBundleCloseBundleResourceMap

Closes an open resource map for a bundle.

```
void CFBundleCloseBundleResourceMap (
   CFBundleRef bundle,
   CFBundleRefNum refNum
);
```

Parameters

bundle

The bundle whose resource map is referenced by *refNum*.

refNum

The reference number for a resource map to close.

Discussion

You open a resource map using either `CFBundleOpenBundleResourceFiles` (page 37) or `CFBundleOpenBundleResourceMap` (page 38).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleCopyAuxiliaryExecutableURL

Returns the location of a bundle's auxiliary executable code.

```
CFURLRef CFBundleCopyAuxiliaryExecutableURL (
   CFBundleRef bundle,
   CFStringRef executableName
);
```

Parameters

bundle

The bundle to examine.

executableName

The name of *bundle*'s auxiliary executable code.

Return Value

The URL location of the specified bundle's auxiliary executable code, or `NULL` if it could not be found. Ownership follows the Create Rule.

Discussion

This function can be used to find executables other than your main executable. This is useful, for instance, for applications that have some command line tool that is packaged with and used by the application. The tool can be packaged in the various platform executable directories in the bundle and can be located with this function. This allows an application to ship versions of the tool for each platform as it does for the main application executable.

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

Related Sample Code
BackgroundExporter

BSDLLCTest

MoreIsBetter

QISA

Declared In
`CFBundle.h`

CFBundleCopyBuiltInPlugInsURL

Returns the location of a bundle's built in plug-in.

```
CFURLRef CFBundleCopyBuiltInPlugInsURL (
   CFBundleRef bundle
);
```

Parameters

`bundle`

The bundle to examine.

Return Value
A CFURL object describing the location of *`bundle`*'s built in plug-ins, or `NULL` if it could not be found. Ownership follows the Create Rule.

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

Related Sample Code
BasicPlugIn

PDEProject

QISA

Declared In
`CFBundle.h`

CFBundleCopyBundleLocalizations

Returns an array containing a bundle's localizations.

```
CFArrayRef CFBundleCopyBundleLocalizations (
   CFBundleRef bundle
);
```

Parameters

`bundle`

The bundle to examine.

Return Value
An array containing *`bundle`*'s localizations. Ownership follows the Create Rule.

Discussion

The array returned by this function is typically passed as a parameter to either the `CFBundleCopyPreferredLocalizationsFromArray` (page 17) or `CFBundleCopyLocalizationsForPreferences` (page 15) function.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleCopyBundleURL

Returns the location of a bundle.

```
CFURLRef CFBundleCopyBundleURL (
   CFBundleRef bundle
);
```

Parameters

`bundle`

The bundle to examine.

Return Value

A CFURL object describing the location of *`bundle`*, or `NULL` if the specified bundle does not exist. Ownership follows the Create Rule.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

GrabBag

HID Utilities Source

QISA

simpleJavaLauncher

Declared In

`CFBundle.h`

CFBundleCopyExecutableArchitectures

Returns an array of CFNumbers representing the architectures a given bundle provides.

```
CFArrayRef CFBundleCopyExecutableArchitectures (
   CFBundleRef bundle
);
```

Parameters

`bundle`

The bundle to examine.

Return Value
If the bundle's executable exists and is a Mach-o file, returns an array of CFNumbers whose values are integers representing the architectures the file provides. Possible values are listed in "Architecture Types" (page 44). If the executable is not a Mach-o file, returns `NULL`. Ownership follows the Create Rule.

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

See Also
`CFBundleCopyExecutableArchitecturesForURL` (page 13)

Declared In
`CFBundle.h`

CFBundleCopyExecutableArchitecturesForURL

Returns an array of CFNumbers representing the architectures a given URL provides.

```
CFArrayRef CFBundleCopyExecutableArchitecturesForURL (
   CFURLRef url
);
```

Parameters

`url`
The URL to examine.

Return Value
For a directory URL, if the bundle's executable exists and is a Mach-o file, returns an array of CFNumbers whose values are integers representing the architectures the URL provides. For a plain file URL representing an unbundled executable, returns the architectures it provides if it is a Mach-o file. Possible values are listed in "Architecture Types" (page 44). If there is no bundle executable or if the executable is not a Mach-o file, returns `NULL`. Ownership follows the Create Rule.

Discussion
For a directory URL, this is equivalent to calling `CFBundleCopyExecutableArchitectures` (page 12) on the corresponding bundle.

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

See Also
`CFBundleCopyExecutableArchitectures` (page 12)

Declared In
`CFBundle.h`

CFBundleCopyExecutableURL

Returns the location of a bundle's main executable code.

```
CFURLRef CFBundleCopyExecutableURL (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFURL object describing the location of *bundle*'s executable code, or `NULL` if none is found. Ownership follows the Create Rule.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CheckExecutableArchitecture

MemoryBasedBundle

Declared In

`CFBundle.h`

CFBundleCopyInfoDictionaryForURL

Returns the information dictionary for a given URL location.

```
CFDictionaryRef CFBundleCopyInfoDictionaryForURL (
   CFURLRef url
);
```

Parameters

url

A CFURL object describing the location of a file.

Return Value

A CFDictionary object containing *url*'s information dictionary. Ownership follows the Create Rule.

Discussion

For a directory URL, this is equivalent to `CFBundleCopyInfoDictionaryInDirectory` (page 14). For a plain file URL representing an unbundled application, this function will attempt to read an information dictionary either from the `(__TEXT, __info_plist)` section of the file (for a Mach-o file) or from a `plst` resource.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CFBundle.h`

CFBundleCopyInfoDictionaryInDirectory

Returns a bundle's information dictionary.

```
CFDictionaryRef CFBundleCopyInfoDictionaryInDirectory (
   CFURLRef bundleURL
);
```

Parameters

bundleURL

A CFURL object describing the location of a bundle.

Return Value

A CFDictionary object containing the information dictionary for a bundle located at *bundleURL*. Ownership follows the Create Rule.

Discussion

This function provides a means to obtain an information dictionary for a bundle without first creating a CFBundle object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleCopyLocalizationsForPreferences

Given an array of possible localizations and preferred locations, returns the one or more of them that CFBundle would use, without reference to the current application context.

```
CFArrayRef CFBundleCopyLocalizationsForPreferences (
   CFArrayRef locArray,
   CFArrayRef prefArray
);
```

Parameters

locArray

An array of possible localizations to search.

prefArray

An array of preferred localizations. If `NULL`, the user's actual preferred localizations will be used.

Return Value

An array containing the localizations that CFBundle would use. Ownership follows the Create Rule.

Discussion

This is not the same as `CFBundleCopyPreferredLocalizationsFromArray` (page 17), because that function takes the current application context into account. To determine the localizations that another application would use, apply this function to the result of `CFBundleCopyBundleLocalizations` (page 11).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CFBundle.h`

CFBundleCopyLocalizationsForURL

Returns an array containing the localizations for a bundle or executable at a particular location.

```
CFArrayRef CFBundleCopyLocalizationsForURL (
   CFURLRef url
);
```

Parameters

url

The location of a bundle's localizations.

Return Value

An array containing the localizations available at *url*. Ownership follows the Create Rule.

Discussion

For a directory URL, this is equivalent to calling the `CFBundleCopyBundleLocalizations` (page 11) function on the corresponding bundle. For a plain file URL representing an unbundled application, this will attempt to determine its localizations using the `kCFBundleLocalizationsKey` (page 44) and `kCFBundleDevelopmentRegionKey` (page 43) keys in the dictionary returned by `CFBundleCopyInfoDictionaryForURL` (page 14), or a `vers` resource if those are not present.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CFBundle.h`

CFBundleCopyLocalizedString

Returns a localized string from a bundle's strings file.

```
CFStringRef CFBundleCopyLocalizedString (
   CFBundleRef bundle,
   CFStringRef key,
   CFStringRef value,
   CFStringRef tableName
);
```

Parameters

bundle

The bundle to examine.

key

The key for the localized string to retrieve. This key will be used to look up the localized string in the strings file. Typically the key is identical to the value of the localized string in the development language.

value

A default value to return if no value exists for *key*.

tableName

The name of the strings file to search. The name should not include the `strings` filename extension. The case of the string must match that of the file name, even on file systems (such as HFS+) that are not case sensitive with regards to file names

Return Value
A CFString object that contains the localized string. If no value exists for *key*, returns *value* unless *value* is `NULL` or an empty string, in which case *key* is returned instead. Ownership follows the Create Rule.

Discussion
This is the base function from which the other localized string macros are derived. In general you should not use this function because the `genstrings` development tool only recognizes the macro version of this call when generating strings files. See `CFCopyLocalizedString` (page 39) for details on how to use these macros.

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

Related Sample Code
MoreSCF

QISA

Declared In
`CFBundle.h`

CFBundleCopyPreferredLocalizationsFromArray

Given an array of possible localizations, returns the one or more of them that CFBundle would use in the current application context.

```
CFArrayRef CFBundleCopyPreferredLocalizationsFromArray (
   CFArrayRef locArray
);
```

Parameters

locArray

An array of possible localizations.

Return Value
A subset of *locArray* that CFBundle would use in the current application context. Ownership follows the Create Rule.

Discussion
You can obtain *locArray* using the `CFBundleCopyBundleLocalizations` (page 11) function.

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

Declared In
`CFBundle.h`

CFBundleCopyPrivateFrameworksURL

Returns the location of a bundle's private Frameworks directory.

```
CFURLRef CFBundleCopyPrivateFrameworksURL (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFURL object describing the location of *bundle*'s private frameworks directory, or `NULL` if it could not be found. Ownership follows the Create Rule.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

SpellingChecker CarbonCocoa Bundled

SpellingChecker-CarbonCocoa

Declared In

`CFBundle.h`

CFBundleCopyResourcesDirectoryURL

Returns the location of a bundle's Resources directory.

```
CFURLRef CFBundleCopyResourcesDirectoryURL (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFURL object describing the location of *bundle*'s resources directory, or `NULL` if it could not be found. Ownership follows the Create Rule.

Discussion

In general, you should never need to use this function. Use `CFBundleCopyResourceURL` (page 19) and similar functions instead.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AppleScriptRunner

CarbonCocoa_PictureCursor

FileNotification

PBORenderToVertexArray

Declared In

`CFBundle.h`

CFBundleCopyResourceURL

Returns the location of a resource contained in the specified bundle.

```
CFURLRef CFBundleCopyResourceURL (
   CFBundleRef bundle,
   CFStringRef resourceName,
   CFStringRef resourceType,
   CFStringRef subDirName
);
```

Parameters

bundle
: The bundle to examine.

resourceName
: The name of the requested resource.

resourceType
: The abstract type of the requested resource. The type is expressed as a filename extension, such as `jpg`. Pass `NULL` if you don't need to search by type.

subDirName
: The name of the subdirectory of the bundle's resources directory to search. Pass `NULL` to search the standard CFBundle resource locations.

Return Value
A CFURL object describing the location of the requested resource, or `NULL` if the resource cannot be found. Ownership follows the Create Rule.

Discussion
For example, if a bundle contains a subdirectory `WaterSounds` that includes a file `Water1.aiff`, you can retrieve the URL for the file using:

```
CFBundleCopyResourceURL(bundle, CFSTR("Water1"), CFSTR("aiff"),
CFSTR("WaterSounds"));
```

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

Related Sample Code
CoreTextTest

HID Utilities Source

QISA

QuartzCache

SampleDS

Declared In
`CFBundle.h`

CFBundleCopyResourceURLForLocalization

Returns the location of a localized resource in a bundle.

```
CFURLRef CFBundleCopyResourceURLForLocalization (
   CFBundleRef bundle,
   CFStringRef resourceName,
   CFStringRef resourceType,
   CFStringRef subDirName,
   CFStringRef localizationName
);
```

Parameters

bundle

The bundle to examine.

resourceName

The name of the requested resource.

resourceType

The abstract type of the resource to locate. The type is expressed as a filename extension, such as `jpg`.

subDirName

The name of the subdirectory of the bundle's resources directory to search. Pass `NULL` to search the standard CFBundle resource locations.

localizationName

The name of the localization. This value should correspond to the name of one of the bundle's language-specific resource directories without the `.lproj` extension. (This parameter is treated literally: If you pass `"de"`, the function will not match resources in a `German.lproj` directory in the bundle.)

Return Value

The location of a localized resource in *bundle*, or `NULL` if the resource could not be found. Ownership follows the Create Rule.

Discussion

Note that file names are case-sensitive, even on file systems (such as HFS+) that are not case sensitive with regards to file names.

You should typically have little reason to use this function (see Getting the Current Language and Locale)—CFBundle's interfaces automatically apply the user's preferences to determine which localized resource files to return in response to a programmatic request. See also `CFBundleCopyBundleLocalizations` (page 11) for how to determine what localizations are available

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleCopyResourceURLInDirectory

Returns the location of a resource contained in the specified bundle directory without requiring the creation of a CFBundle object.

```
CFURLRef CFBundleCopyResourceURLInDirectory (
   CFURLRef bundleURL,
   CFStringRef resourceName,
   CFStringRef resourceType,
   CFStringRef subDirName
);
```

Parameters

bundleURL

The bundle to examine.

resourceName

The name of the requested resource.

resourceType

The abstract type of the requested resource. The type is expressed as a filename extension, such as `jpg`. Pass `NULL` if you don't need to search by type.

subDirName

The name of the subdirectory of the bundle's resources directory to search. Pass `NULL` to search the standard CFBundle resource locations.

Return Value

A CFURL object describing the location of the requested resource, or `NULL` if the resource cannot be found. Ownership follows the Create Rule.

Discussion

This function provides a means to obtain package information for a bundle without first creating a bundle. However, since CFBundle objects cache search results, it is faster to create a CFBundle object if you need to repeatedly access resources.

Note that searches are case-sensitive, even on file systems (such as HFS+) that are not case sensitive with regards to file names.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleCopyResourceURLsOfType

Assembles an array of URLs specifying all of the resources of the specified type found in a bundle.

```
CFArrayRef CFBundleCopyResourceURLsOfType (
   CFBundleRef bundle,
   CFStringRef resourceType,
   CFStringRef subDirName
);
```

Parameters

bundle

The bundle to examine.

resourceType

The abstract type of the resources to locate. The type is expressed as a filename extension, such as `jpg`.

subDirName

The name of the subdirectory of the bundle’s resources directory to search. Pass `NULL` to search the standard CFBundle resource locations.

Return Value

A CFArray object containing CFURL objects of the requested resources. Ownership follows the Create Rule.

Discussion

Note that searches are case-sensitive, even on file systems (such as HFS+) that are not case sensitive with regards to file names.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Carbon Porting Tutorial

Declared In

`CFBundle.h`

CFBundleCopyResourceURLsOfTypeForLocalization

Returns an array containing copies of the URL locations for a specified bundle, resource, and localization name.

```
CFArrayRef CFBundleCopyResourceURLsOfTypeForLocalization (
   CFBundleRef bundle,
   CFStringRef resourceType,
   CFStringRef subDirName,
   CFStringRef localizationName
);
```

Parameters

bundle

The bundle to examine.

resourceType

The abstract type of the resources to locate. The type is expressed as a filename extension, such as `jpg`.

subDirName

The name of the subdirectory of the bundle’s Resources directory to search. Pass `NULL` to search the standard CFBundle resource locations.

localizationName

The name of the localization. This value should correspond to the name of one of the bundle's language-specific resource directories without the `.lproj` extension. (This parameter is treated literally: If you pass `"de"`, the function will not match resources in a `German.lproj` directory in the bundle.)

Return Value

A CFArray object containing copies of the requested locations. Ownership follows the Create Rule.

Discussion

Note that file names are case-sensitive, even on file systems (such as HFS+) that are not case sensitive with regards to file names.

You should typically have little reason to use this function (see Getting the Current Language and Locale)—CFBundle's interfaces automatically apply the user's preferences to determine which localized resource files to return in response to a programmatic request. See also `CFBundleCopyBundleLocalizations` (page 11) for how to determine what localizations are available

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

Declared In
`CFBundle.h`

CFBundleCopyResourceURLsOfTypeInDirectory

Returns an array of CFURL objects describing the locations of all resources in a bundle of the specified type without needing to create a CFBundle object.

```
CFArrayRef CFBundleCopyResourceURLsOfTypeInDirectory (
   CFURLRef bundleURL,
   CFStringRef resourceType,
   CFStringRef subDirName
);
```

Parameters

bundleURL
: The location of a bundle to examine.

resourceType
: The abstract type of the resources to locate. The type is expressed as a filename extension, such as `jpg`.

subDirName
: The name of the subdirectory of the bundle's resources directory to search. Pass `NULL` to search the standard CFBundle resource locations.

Return Value
A CFArray object containing the CFURL objects of the requested resources. Ownership follows the Create Rule.

Discussion
This function provides a means to obtain an array containing the locations of all of the requested resources without first creating a CFBundle object. However, since CFBundle objects cache search results, it is faster to create a CFBundle object if you need to repeatedly access resources.

Note that file names are case-sensitive, even on file systems (such as HFS+) that are not case sensitive with regards to file names.

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

Declared In
`CFBundle.h`

CFBundleCopySharedFrameworksURL

Returns the location of a bundle's shared frameworks directory.

```
CFURLRef CFBundleCopySharedFrameworksURL (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFURL object containing the location of *bundle*'s shared frameworks directory, or `NULL` if it could not be found. Ownership follows the Create Rule.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleCopySharedSupportURL

Returns the location of a bundle's shared support files directory.

```
CFURLRef CFBundleCopySharedSupportURL (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFURL object containing the location of *bundle*'s shared support files directory, or `NULL` if it could not be found. Ownership follows the Create Rule.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

BasicInputMethod

Declared In

`CFBundle.h`

CFBundleCopySupportFilesDirectoryURL

Returns the location of the bundle's support files directory.

```
CFURLRef CFBundleCopySupportFilesDirectoryURL (
   CFBundleRef bundle
);
```

Parameters

bundle

The CFBundle object whose support files directory you want to locate.

Return Value
A CFURL object describing the location of the bundle's support files directory, or `NULL` if it could not be found. Ownership follows the Create Rule.

Discussion
In general, you should never need to use this function. Use `CFBundleCopyResourceURL` (page 19) and similar functions instead.

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

Declared In
`CFBundle.h`

CFBundleCreate

Creates a CFBundle object.

```
CFBundleRef CFBundleCreate (
   CFAllocatorRef allocator,
   CFURLRef bundleURL
);
```

Parameters

allocator
The allocator to use to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

bundleURL
The location of the bundle for which to create a CFBundle object.

Return Value
A CFBundle object created from the bundle at *bundleURL*. Ownership follows the Create Rule.

Returns `NULL` if there was a memory allocation problem. May return an existing CFBundle object with the reference count incremented. May return `NULL` if the bundle doesn't exist at *bundleURL* (see Discussion).

Discussion
Once a bundle has been created, it is cached; the bundle cache is flushed only periodically. `CFBundleCreate` does not check that a cached bundle still exists in the filesystem. If a bundle is deleted from the filesystem, it is therefore possible for `CFBundleCreate` to return a cached bundle that has actually been deleted.

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

Related Sample Code
MemoryBasedBundle

MoreIsBetter

QISA

SpellingChecker CarbonCocoa Bundled

SpellingChecker-CarbonCocoa

Declared In
`CFBundle.h`

CFBundleCreateBundlesFromDirectory

Searches a directory and constructs an array of CFBundle objects from all valid bundles in the specified directory.

```
CFArrayRef CFBundleCreateBundlesFromDirectory (
   CFAllocatorRef allocator,
   CFURLRef directoryURL,
   CFStringRef bundleType
);
```

Parameters

allocator

The allocator to use to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

directoryURL

The location of the directory to search for valid bundles.

bundleType

The abstract type of the bundles to locate and create. The type is expressed as a filename extension, such as `bundle`. Pass `NULL` to create CFBundle objects for bundles of any type.

Return Value

A CFArray object containing CFBundle objects created from the contents of the specified directory. Returns an empty array if no bundles exist at *directoryURL*, and `NULL` if there was a memory allocation problem. Ownership follows the Create Rule.

Discussion

The array returned by this function will not contain stale CFBundle references.

Special Considerations

The Create Rule applies both to the array returned and to the bundles in the array. In order to properly dispose of the returned value, you must release the array *and* any bundles returned in the array.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleGetAllBundles

Returns an array containing all of the bundles currently open in the application.

```
CFArrayRef CFBundleGetAllBundles (
   void
);
```

Return Value

A CFArray object containing CFBundle objects for each open bundle in the application. Ownership follows the Get Rule.

Discussion

This function is potentially expensive, so use with care.

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

Declared In
`CFBundle.h`

CFBundleGetBundleWithIdentifier

Locate a bundle given its program-defined identifier.

```
CFBundleRef CFBundleGetBundleWithIdentifier (
   CFStringRef bundleID
);
```

Parameters

`bundleID`
The identifier of the bundle to locate. Note that identifier names are case-sensitive.

Return Value
A CFBundle object, or `NULL` if the bundle was not found. Ownership follows the Get Rule.

Discussion
For a bundle to be located using its identifier, the bundle object must have already been created. The principal intended purpose for locating bundles by identifier is so that code (in frameworks, plugins, etc.) can find its own bundle. If a bundle is created, then the bundle deleted from the filesystem and this function invoked afterwards, it will still return the original bundle.

Bundle identifiers are created by entering a value for the key `CFBundleIdentifier` in the bundle's `Info.plist` file.

To guarantee uniqueness, bundle identifiers take the form of Java style package names, such as `com.MyCompany.MyApp.bundleName`.

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

Related Sample Code
CarbonMDEF

ElectricImageComponent

QTPixelBufferVCToCGImage

SampleCMPlugIn

SampleDS

Declared In
`CFBundle.h`

CFBundleGetDataPointerForName

Returns a data pointer to a symbol of the given name.

```
void * CFBundleGetDataPointerForName (
   CFBundleRef bundle,
   CFStringRef symbolName
);
```

Parameters

bundle

The bundle to examine.

symbolName

The name of the symbol you are searching for.

Return Value

A data pointer to a symbol named *symbolName* in *bundle*, or `NULL` if *symbolName* cannot be found. Ownership follows the Get Rule.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleGetDataPointersForNames

Returns a C array of data pointer to symbols of the given names.

```
void CFBundleGetDataPointersForNames (
   CFBundleRef bundle,
   CFArrayRef symbolNames,
   void *stbl[]
);
```

Parameters

bundle

The bundle to examine.

symbolNames

A CFArray object containing CFString objects representing the symbol names to search for.

stbl

A C array into which this function stores the data pointers for the symbols specified in *symbolNames*. The array contains `NULL` for any names in *symbolNames* that cannot be found.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleGetDevelopmentRegion

Returns the bundle's development region from the bundle's information property list.

```
CFStringRef CFBundleGetDevelopmentRegion (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFString object containing the name of the bundle's development region. Ownership follows the Get Rule.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleGetFunctionPointerForName

Returns a pointer to a function in a bundle's executable code using the function name as the search key.

```
void * CFBundleGetFunctionPointerForName (
   CFBundleRef bundle,
   CFStringRef functionName
);
```

Parameters

bundle

The bundle to examine.

functionName

The name of the function to locate.

Return Value

A pointer to a function in a *bundle*'s executable code, or `NULL` if *functionName* cannot be found. Ownership follows the Get Rule.

Discussion

Calling this function will cause the bundle's code to be loaded if necessary.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

PDEProject

QISA

SpellingChecker CarbonCocoa Bundled

SpellingChecker-CarbonCocoa

VideoProcessing

Declared In

`CFBundle.h`

CFBundleGetFunctionPointersForNames

Constructs a function table containing pointers to all of the functions found in a bundle's main executable code.

```
void CFBundleGetFunctionPointersForNames (
   CFBundleRef bundle,
   CFArrayRef functionNames,
   void *ftbl[]
);
```

Parameters

bundle

The bundle to examine.

functionNames

A CFArray object containing a list of the function names to locate.

ftbl

A C array into which this function stores the function pointers for the symbols specified in *functionNames*. The array contains `NULL` for any names in *functionNames* that cannot be found.

Discussion

Calling this function causes the bundle's code to be loaded if necessary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleGetIdentifier

Returns the bundle identifier from a bundle's information property list.

```
CFStringRef CFBundleGetIdentifier (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value

A CFString object containing the bundle's identifier, or `NULL` if none was specified in the information property list. Ownership follows the Get Rule.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CFPrefTopScores

Declared In

`CFBundle.h`

CFBundleGetInfoDictionary

Returns a bundle's information dictionary.

```
CFDictionaryRef CFBundleGetInfoDictionary (
   CFBundleRef bundle
);
```

Parameters

bundle
The bundle to examine.

Return Value

A CFDictionary object containing the data stored in the bundle's information property list (the `Info.plist` file). This is a global information dictionary. CFBundle may add extra keys to this dictionary for its own use. Ownership follows the Get Rule.

Discussion

You should typically use `CFBundleGetValueForInfoDictionaryKey` (page 34) rather than retrieving values directly from the info dictionary because the function will return localized values if any are available. Use `CFBundleGetInfoDictionary` only if you know that the key you are interested in will not be localized.

To retrieve an info dictionary without creating a CFBundle object, see `CFBundleCopyInfoDictionaryInDirectory` (page 14) and `CFBundleCopyInfoDictionaryForURL` (page 14).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

BSDLLCTest

Declared In

`CFBundle.h`

CFBundleGetLocalInfoDictionary

Returns a bundle's localized information dictionary.

```
CFDictionaryRef CFBundleGetLocalInfoDictionary (
   CFBundleRef bundle
);
```

Parameters

bundle
The bundle to examine.

Return Value

A dictionary containing the key-value pairs in *bundle*'s localized information dictionary (from the `InfoPlist.strings` file for the current locale). Ownership follows the Get Rule.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

VertexPerformanceTest

Declared In
`CFBundle.h`

CFBundleGetMainBundle

Returns an application's main bundle.

```
CFBundleRef CFBundleGetMainBundle (
   void
);
```

Return Value
A CFBundle object representing the application's main bundle, or `NULL` if it is not possible to create a bundle. Ownership follows the Get Rule.

Discussion
CFBundle creates a main bundle whenever it possibly can, even for unbundled apps. There are a few situations in which it is not possible, so you should check the return value against `NULL`, but this happens only in exceptional circumstances.

For an explanation of the main bundle, see Locating and Opening Bundles in *Bundle Programming Guide*.

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

Related Sample Code
BSDLLCTest

Carbon Porting Tutorial

HID Utilities Source

MoreIsBetter

QISA

Declared In
`CFBundle.h`

CFBundleGetPackageInfo

Returns a bundle's package type and creator.

```
void CFBundleGetPackageInfo (
   CFBundleRef bundle,
   UInt32 *packageType,
   UInt32 *packageCreator
);
```

Parameters

`bundle`
: The bundle to examine.

`packageType`
: On return, the four-letter Mac OS-style type code for the bundle. This is `APPL` for applications, `FMWK` for frameworks, and `BNDL` for generic bundles. Or a more specific type code for generic bundles.

packageCreator

On return, the four-letter Mac OS-style "creator" code for the bundle.

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

Declared In
`CFBundle.h`

CFBundleGetPackageInfoInDirectory

Returns a bundle's package type and creator without having to create a CFBundle object.

```
Boolean CFBundleGetPackageInfoInDirectory (
   CFURLRef url,
   UInt32 *packageType,
   UInt32 *packageCreator
);
```

Parameters

url

The location of a bundle.

packageType

On return, the four-letter Mac OS-style type code for the bundle. This is `APPL` for applications, `FMWK` for frameworks, and `BNDL` for generic bundles. Or a more specific type code for generic bundles.

packageCreator

On return, the four-letter Mac OS-style "creator" code for the bundle.

Return Value
`true` if the package type and creator were found, otherwise `false`.

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

Declared In
`CFBundle.h`

CFBundleGetPlugIn

Returns a bundle's plug-in.

```
CFPlugInRef CFBundleGetPlugIn (
   CFBundleRef bundle
);
```

Parameters

bundle

The bundle to examine.

Return Value
The plug-in for *bundle*. Ownership follows the Get Rule.

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

Declared In
`CFBundle.h`

CFBundleGetTypeID

Returns the type identifier for the CFBundle opaque type.

```
CFTypeID CFBundleGetTypeID (
   void
);
```

Return Value
The type identifier for the CFBundle opaque type.

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

Declared In
`CFBundle.h`

CFBundleGetValueForInfoDictionaryKey

Returns a value (localized if possible) from a bundle's information dictionary.

```
CFTypeRef CFBundleGetValueForInfoDictionaryKey (
   CFBundleRef bundle,
   CFStringRef key
);
```

Parameters

bundle
The bundle to examine.

key
The key for the value to return.

Return Value
A value corresponding to *key* in *bundle*'s information dictionary. If available, a localized value is returned, otherwise the global value is returned. Ownership follows the Get Rule.

Discussion
You should use this function rather than retrieving values directly from the info dictionary (`Info.plist`) because `CFBundleGetValueForInfoDictionaryKey` returns localized values if any are available (from the `InfoPlist.strings` file for the current locale).

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

Related Sample Code
CFPreferences
GrabBag
PDEProject
SampleCMPlugIn

Declared In
`CFBundle.h`

CFBundleGetVersionNumber

Returns a bundle's version number.

```
UInt32 CFBundleGetVersionNumber (
   CFBundleRef bundle
);
```

Parameters

`bundle`
: The bundle to examine. The bundle's version number can be number or a string of the standard form "2.5.3d5".

Return Value
A Mac OS `vers` resource style version number. If the bundle's version number is a number, it is interpreted as the unsigned long integer format defined by the `vers` resource on Mac OS 9. If it is a string, it is automatically converted to the numeric representation, where the major version number is restricted to 2 BCD digits (in other words, it must be in the range 0-99) and the minor and bug fix version numbers are each restricted to a single BCD digit (0-9). See Technical Note TN1132 for more details.

Discussion
This function is only supported for the Mac OS `vers` resource style version numbers. Where other version number styles—namely X, or X.Y, or X.Y.Z—are used, you can use `CFBundleGetValueForInfoDictionaryKey` (page 34) with the key `kCFBundleVersionKey` to extract the version number as a string from the bundle's information dictionary.

Some version numbers of the form X, X.Y, and X.Y.Z may work with this function, if X <= 99, Y <= 9, and Z <= 9. Thus a version number 76.5.4 will work, but 76.12 will not work.

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

Related Sample Code
iTunesController

Declared In
`CFBundle.h`

CFBundleIsExecutableLoaded

Obtains information about the load status for a bundle's main executable.

```
Boolean CFBundleIsExecutableLoaded (
   CFBundleRef bundle
);
```

Parameters

`bundle`
: The bundle to examine.

Return Value
`true` if *`bundle`*'s main executable has been loaded, otherwise `false`.

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

Related Sample Code
VideoProcessing

Declared In
`CFBundle.h`

CFBundleLoadExecutable

Loads a bundle's main executable code into memory and dynamically links it into the running application.

```
Boolean CFBundleLoadExecutable (
   CFBundleRef bundle
);
```

Parameters

`bundle`
The bundle whose main executable you want to load.

Return Value
`true` if the executable was successfully loaded, otherwise `false`.

Discussion
You should typically try to avoid using this function, but instead use `CFBundleGetFunctionPointerForName` (page 29) and related functions since these make memory management of the bundle easier.

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

Related Sample Code
MoreIsBetter
QISA
SpellingChecker CarbonCocoa Bundled
SpellingChecker-CarbonCocoa
VideoProcessing

Declared In
`CFBundle.h`

CFBundleLoadExecutableAndReturnError

Returns a Boolean value that indicates whether a given bundle is loaded, attempting to load it if necessary.

```
Boolean CFBundleLoadExecutableAndReturnError (
   CFBundleRef bundle,
   CFErrorRef *error
);
```

Parameters

bundle

The bundle to examine.

error

Upon return, if an error occurs contains a CFError that describes the problem. Ownership follows the Create Rule.

Return Value

If *bundle* is already loaded, returns `true`. If *bundle* is not already loaded, attempts to load *bundle*; if that attempt succeeds returns `true`, otherwise returns `false`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CFBundle.h`

CFBundleOpenBundleResourceFiles

Opens the non-localized and localized resource files (if any) for a bundle in separate resource maps.

```
SInt32 CFBundleOpenBundleResourceFiles (
   CFBundleRef bundle,
   CFBundleRefNum *refNum,
   CFBundleRefNum *localizedRefNum
);
```

Parameters

bundle

The bundle whose resource map you want to open.

refNum

On return, the reference number of the non-localized resource map.

localizedRefNum

On return, the reference number of the localized resource map.

Return Value

An error code. The function returns `0` (`noErr`) if successful. If the bundle contains more than one resource file, the function returns an error code only if none was opened. The most common error is `resFNotFound`, but the function may also pass through other errors returned from the Resource Manager.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBundle.h`

CFBundleOpenBundleResourceMap

Opens the non-localized and localized resource files (if any) for a bundle in a single resource map.

```
CFBundleRefNum CFBundleOpenBundleResourceMap (
   CFBundleRef bundle
);
```

Parameters

`bundle`

The bundle whose resource map you want to open.

Return Value

A distinct reference number for the resource map.

Discussion

Creates and makes current a single read-only resource map containing the non-localized and localized resource files. If this function is called multiple times, it opens the files multiple times and returns distinct reference numbers for each. Use `CFBundleCloseBundleResourceMap` (page 10) to close a resource map.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

SampleDS

Declared In

`CFBundle.h`

CFBundlePreflightExecutable

Returns a Boolean value that indicates whether a given bundle is loaded or appears to be loadable.

```
Boolean CFBundlePreflightExecutable (
   CFBundleRef bundle,
   CFErrorRef *error
);
```

Parameters

`bundle`

The bundle to examine.

`error`

Upon return, if an error occurs contains a CFError that describes the problem. Ownership follows the Create Rule.

Return Value

`true` if *`bundle`* is loaded or upon inspection appears to be loadable, otherwise `false`.

Discussion

If this function returns true, this does not mean that the bundle is definitively loadable, since it may fail to load due to link errors or other problems not readily detectable.

Availability

Available in Mac OS X v10.5 and later.

Declared In
`CFBundle.h`

CFBundleUnloadExecutable

Unloads the main executable for the specified bundle.

```
void CFBundleUnloadExecutable (
   CFBundleRef bundle
);
```

Parameters

`bundle`
    The bundle whose main executable you want to unload.

Discussion
You should typically try to avoid using this function, but instead use `CFBundleGetFunctionPointerForName` (page 29) and related functions since these make management of the bundle easier (when the last reference to the CFBundle object is released, and it is finally deallocated, then the code will be unloaded if it is still loaded, and if the executable is of a type that supports unloading).

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

Related Sample Code
CFM_MachO_CFM

HID Utilities Source

Declared In
`CFBundle.h`

CFCopyLocalizedString

Searches the default strings file `Localizable.strings` for the string associated with the specified key.

```
CFStringRef CFCopyLocalizedString (
    CFStringRef key,
    const char *comment
);
```

Parameters

`key`
    The development language version of the string. This string is used as the search key to locate the localized version of the string.

`comment`
    A comment to provide the translators with contextual information necessary for proper translation.

Return Value
The localized version of the requested string. Returns *`key`* if no value corresponding to *`key`* is found. Ownership follows the Create Rule.

Discussion
This is a macro variant of `CFBundleCopyLocalizedString` (page 16) for use with the `genstrings` tool.

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

Related Sample Code
Cocoa PDE with Carbon Printing

PDEProject

QISA

Quartz2DBasics

TypeServicesForUnicode

Declared In
`CFBundle.h`

CFCopyLocalizedStringFromTable

Searches the specified strings file for the string associated with the specified key.

```
CFStringRef CFCopyLocalizedStringFromTable (
    CFStringRef key,
    CFStringRef tableName,
    const char *comment
);
```

Parameters

`key`

The development language version of the string. This string is used as the search key to locate the localized version of the string.

`tableName`

The name of the strings file to search. The name should not include the `strings` filename extension. The case of the string must match that of the file name, even on file systems (such as HFS+) that are not case sensitive with regards to file names

`comment`

A comment to provide the translators with contextual information necessary for proper translation.

Return Value
The localized version of the requested string, or *`key`* if no value corresponding to *`key`* is found. Ownership follows the Create Rule.

Discussion
This is a macro variant of `CFBundleCopyLocalizedString` (page 16) for use with the `genstrings` tool.

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

Declared In
`CFBundle.h`

CFCopyLocalizedStringFromTableInBundle

Returns a localized version of the specified string.

```
CFStringRef CFCopyLocalizedStringFromTableInBundle (
    CFStringRef key,
    CFStringRef tableName,
    CFBundleRef bundle,
    const char *comment
);
```

Parameters

key

The development language version of the string. This string is used as the search key to locate the localized version of the string.

tableName

The name of the strings file to search. The name should not include the `strings` filename extension. The case of the string must match that of the file name, even on file systems (such as HFS+) that are not case sensitive with regards to file names

bundle

The bundle to examine.

comment

A comment to provide the translators with contextual information necessary for proper translation.

Return Value

The localized version of the requested string, or *key* if no value corresponding to *key* is found. Ownership follows the Create Rule.

Discussion

This is a macro variant of `CFBundleCopyLocalizedString` (page 16) for use with the `genstrings` tool.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Carbon Porting Tutorial

PDEProject

Declared In

`CFBundle.h`

CFCopyLocalizedStringWithDefaultValue

Returns a localized version of a localization string.

```
CFStringRef CFCopyLocalizedStringWithDefaultValue (
    CFStringRef key,
    CFStringRef tableName,
    CFBundleRef bundle,
    CFStringRef value,
    const char *comment
);
```

Parameters

key

The development language version of the string. This string is used as the search key to locate the localized version of the string.

tableName
: The name of the strings file to search. The name should not include the `strings` filename extension.

bundle
: The bundle to examine.

value
: The default value for the requested localization string.

comment
: A comment to provide the translators with contextual information necessary for proper translation.

Return Value
The localized version of the requested string, or *key* if no value corresponding to *key* is found. Ownership follows the Create Rule.

Discussion
This is a macro variant of `CFBundleCopyLocalizedString` (page 16) for use with the `genstrings` tool.

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

Declared In
`CFBundle.h`

Data Types

CFBundleRef

A reference to a CFBundle object.

```
typedef struct __CFBundle *CFBundleRef;
```

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

Declared In
`CFBundle.h`

CFBundleRefNum

Type that identifies a distinct reference number for a resource map.

```
#if __LP64__
typedef int CFBundleRefNum;
#else /* __LP64__ */
typedef SInt16 CFBundleRefNum;
#endif /* __LP64__ */
```

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

Declared In
`CFBundle.h`

Constants

Information Property List Keys

Standard keys found in a bundle's information property list file.

```
const CFStringRef kCFBundleInfoDictionaryVersionKey;
const CFStringRef kCFBundleExecutableKey;
const CFStringRef kCFBundleIdentifierKey;
const CFStringRef kCFBundleVersionKey;
const CFStringRef kCFBundleDevelopmentRegionKey;
const CFStringRef kCFBundleNameKey;
const CFStringRef kCFBundleLocalizationsKey;
```

Constants

`kCFBundleInfoDictionaryVersionKey`

The version of the information property list format.

Available in Mac OS X v10.0 and later.

Declared in `CFBundle.h`.

`kCFBundleExecutableKey`

The name of the executable in this bundle (if any).

Available in Mac OS X v10.0 and later.

Declared in `CFBundle.h`.

`kCFBundleIdentifierKey`

The bundle identifier.

Available in Mac OS X v10.0 and later.

Declared in `CFBundle.h`.

`kCFBundleVersionKey`

The version number of the bundle.

For Mac OS 9 style version numbers (for example "2.5.3d5"), clients can use `CFBundleGetVersionNumber` (page 35) instead of accessing this key directly since that function will properly convert the version string into its compact integer representation.

Available in Mac OS X v10.0 and later.

Declared in `CFBundle.h`.

`kCFBundleDevelopmentRegionKey`

The name of the development language of the bundle.

When CFBundle looks for resources, the fallback is to look in the lproj whose name is given by the `kCFBundleDevelopmentRegionKey` in the `Info.plist` file. You must, therefore, ensure that a bundle contains an lproj with that exact name containing a copy of every localized resource, otherwise CFBundle cannot guarantee the fallback mechanism will work.

Available in Mac OS X v10.0 and later.

Declared in `CFBundle.h`.

`kCFBundleNameKey`

The human-readable name of the bundle.

This key is often found in the `InfoPlist.strings` since it is usually localized.

Available in Mac OS X v10.0 and later.

Declared in `CFBundle.h`.

`kCFBundleLocalizationsKey`

Allows an unbundled application that handles localization itself to specify which localizations it has available.

Available in Mac OS X v10.2 and later.

Declared in `CFBundle.h`.

Declared In
`CFBundle.h`

Architecture Types

Constants that identify executable architecture types.

```
enum {
    kCFBundleExecutableArchitectureI386     = 0x00000007,
    kCFBundleExecutableArchitecturePPC      = 0x00000012,
    kCFBundleExecutableArchitectureX86_64   = 0x01000007,
    kCFBundleExecutableArchitecturePPC64    = 0x01000012
};
```

Constants

`kCFBundleExecutableArchitectureI386`

Specifies the 32-bit Intel architecture.

Available in Mac OS X v10.5 and later.

Declared in `CFBundle.h`.

`kCFBundleExecutableArchitecturePPC`

Specifies the 32-bit PowerPC architecture.

Available in Mac OS X v10.5 and later.

Declared in `CFBundle.h`.

`kCFBundleExecutableArchitectureX86_64`

Specifies the 64-bit Intel architecture.

Available in Mac OS X v10.5 and later.

Declared in `CFBundle.h`.

`kCFBundleExecutableArchitecturePPC64`

Specifies the 64-bit PowerPC architecture.

Available in Mac OS X v10.5 and later.

Declared in `CFBundle.h`.

Declared In
`CFBundle.h`

Document Revision History

This table describes the changes to *CFBundle Reference*.

Date	Notes
2007-10-18	Clarified the discussion of functions related to getting information from the info dictionary.
2007-10-31	Corrected the return value for the CFBundleCreateBundlesFromDirectory function.
2007-02-26	Updated to include API introduced in Mac OS X v10.5.
2007-02-08	Clarified the return value of the CFBundleCreateBundlesFromDirectory function.
2006-06-28	Enhanced the discussion of bundle loading and unloading in the Introduction, and enhanced the descriptions of CFBundleGetInfoDictionary and kCFBundleDevelopmentRegionKey.
2006-01-10	Described a possible issue concerning a value that is returned from the CFBundleCreate method.
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-08-11	Corrected definition of value parameter in CFBundleCopyLocalizedString, and various minor typographical errors.
2005-04-29	Moved Introduction to new Introduction page.
	Corrected parameter descriptions of `CFBundleCopyLocalizedString` (page 16).
2004-08-31	Minor update to note warning about constant strings when unloading bundles.
	Added notes about case-sensitivity of file names.
	Clarified return values from `CFBundleOpenBundleResourceFiles` (page 37).
2004-06-28	Minor update to clarify version number format used with `CFBundleGetVersionNumber` (page 35).
2003-01-01	First version of this document.

Index