CFType Reference

Core Foundation

2007-07-12

Contents

CFType Reference

Derived From:	None
Framework:	CoreFoundation/CoreFoundation.h
Declared in	CFBase.h
	CFString.h
Companion guides	Core Foundation Design Concepts
	Memory Management Programming Guide for Core Foundation

Overview

All other Core Foundation opaque types derive from CFType. The functions, callbacks, data types, and constants defined for CFType can be used by any derived opaque type. Hence, CFType functions are referred to as "polymorphic functions." You use CFType functions to retain and release objects, to compare and inspect objects, get descriptions of objects and opaque types, and to get object allocators.

Functions by Task

Memory Management

CFGetAllocator (page 8)
> Returns the allocator used to allocate a Core Foundation object.

CFGetRetainCount (page 8)
> Returns the reference count of a Core Foundation object.

CFMakeCollectable (page 10)
> Makes a newly-allocated Core Foundation object eligible for garbage collection.

CFRelease (page 11)
> Releases a Core Foundation object.

CFRetain (page 11)
> Retains a Core Foundation object.

Determining Equality

CFEqual (page 7)
> Determines whether two Core Foundation objects are considered equal.

Hashing

`CFHash` (page 10)

> Returns a code that can be used to identify an object in a hashing structure.

Miscellaneous Functions

`CFCopyDescription` (page 6)

> Returns a textual description of a Core Foundation object.

`CFCopyTypeIDDescription` (page 7)

> Returns a textual description of a Core Foundation type, as identified by its type ID, which can be used when debugging.

`CFGetTypeID` (page 9)

> Returns the unique identifier of an opaque type to which a Core Foundation object belongs.

`CFShow` (page 12)

> Prints a description of a Core Foundation object to stderr.

Functions

CFCopyDescription

Returns a textual description of a Core Foundation object.

```
CFStringRef CFCopyDescription (
    CFTypeRef cf
);
```

Parameters

cf

> The CFType object (a generic reference of type `CFTypeRef` (page 14)) from which to derive a description.

Return Value

A string that contains a description of `cf`. Ownership follows the Create Rule.

Discussion

The nature of the description differs by object. For example, a description of a CFArray object would include descriptions of each of the elements in the collection.

You can use this function for debugging Core Foundation objects in your code. Note, however, that the description for a given object may be different in different releases of the operating system. Do *not* create dependencies in your code on the content or format of the information returned by this function.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CFPrefsDumper

FSFileOperation

MoreIsBetter

MoreSCF

Declared In
CFBase.h

CFCopyTypeIDDescription

Returns a textual description of a Core Foundation type, as identified by its type ID, which can be used when debugging.

```
CFStringRef CFCopyTypeIDDescription (
   CFTypeID type_id
);
```

Parameters

theType

An integer of type CFTypeID (page 13) that uniquely identifies a Core Foundation opaque type.

Return Value

A string containing a type description. Ownership follows the Create Rule.

Discussion

You can use this function for debugging Core Foundation objects in your code. Note, however, that the description for a given object may be different in different releases of the operating system. Do *not* create dependencies in your code on the content or format of the information returned by this function.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MoreIsBetter

MoreSCF

Declared In

CFBase.h

CFEqual

Determines whether two Core Foundation objects are considered equal.

```
Boolean CFEqual (
   CFTypeRef cf1,
   CFTypeRef cf2
);
```

Parameters

cf1

A CFType object to compare to *cf2*.

cf2

A CFType object to compare to *cf1*.

Return Value

`true` if `cf1` and `cf2` are of the same type and considered equal, otherwise `false`.

Discussion

Equality is something specific to each Core Foundation opaque type. For example, two CFNumber objects are equal if the numeric values they represent are equal. Two CFString objects are equal if they represent identical sequences of characters, regardless of encoding.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

bulkerase

databurntest

MoreIsBetter

MoreSCF

QISA

Declared In

`CFBase.h`

CFGetAllocator

Returns the allocator used to allocate a Core Foundation object.

```
CFAllocatorRef CFGetAllocator (
   CFTypeRef cf
);
```

Parameters

`cf`

> The CFType object to examine.

Return Value

The allocator used to allocate memory for `cf`.

Discussion

When you are creating a Core Foundation object sometimes you want to ensure that the block of memory allocated for the object is from the same allocator used for another object. One way to do this is to reuse the allocator assigned to an existing Core Foundation object when you call a "creation" function.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFBase.h`

CFGetRetainCount

Returns the reference count of a Core Foundation object.

```
CFIndex CFGetRetainCount (
    CFTypeRef cf
);
```

Parameters

cf

> The CFType object to examine.

Return Value

A number representing the reference count of *cf*.

Discussion

You increment the reference count using the CFRetain (page 11) function, and decrement the reference count using the CFRelease (page 11) function.

This function may useful for debugging memory leaks. You normally do not use this function, otherwise.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AuthForAll

BSDLLCTest

MoreIsBetter

MoreSCF

QISA

Declared In

CFBase.h

CFGetTypeID

Returns the unique identifier of an opaque type to which a Core Foundation object belongs.

```
CFTypeID CFGetTypeID (
    CFTypeRef cf
);
```

Parameters

cf

> The CFType object to examine.

Return Value

A value of type CFTypeID (page 13) that identifies the opaque type of *cf*.

Discussion

This function returns a value that uniquely identifies the opaque type of any Core Foundation object. You can compare this value with the known CFTypeID (page 13) identifier obtained with a "GetTypeID" function specific to a type, for example CFDateGetTypeID. These values might change from release to release or platform to platform.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code
BSDLLCTest

CFFTPSample

MoreIsBetter

MoreSCF

QISA

Declared In
`CFBase.h`

CFHash

Returns a code that can be used to identify an object in a hashing structure.

```
CFHashCode CFHash (
    CFTypeRef cf
);
```

Parameters

cf

      A CFType object to examine.

Return Value
An integer of type `CFHashCode` (page 13) that represents a hashing value for *cf*.

Discussion
Two objects that are equal (as determined by the `CFEqual` (page 7) function) have the same hashing value. However, the converse is not true: two objects with the same hashing value might not be equal. That is, hashing values are not necessarily unique.

The hashing value for an object might change from release to release or from platform to platform.

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

Declared In
`CFBase.h`

CFMakeCollectable

Makes a newly-allocated Core Foundation object eligible for garbage collection.

```
CFTypeRef CFMakeCollectable (
    CFTypeRef cf
);
```

Parameters

cf

      A CFType object to make collectable. This value must not be `NULL`.

Return Value
cf.

Discussion

For more details, see *Garbage Collection Programming Guide*.

Special Considerations

If `cf` is `NULL`, this will cause a runtime error and your application will crash.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

AutomatorHandsOn

Declared In

`CFBase.h`

CFRelease

Releases a Core Foundation object.

```
void CFRelease (
    CFTypeRef cf
);
```

Parameters

`cf`

A CFType object to release. This value must not be `NULL`.

Discussion

If the retain count of `cf` becomes zero the memory allocated to the object is deallocated and the object is destroyed. If you create, copy, or explicitly retain (see the `CFRetain` (page 11) function) a Core Foundation object, you are responsible for releasing it when you no longer need it (see *Memory Management Programming Guide for Core Foundation*).

Special Considerations

If `cf` is `NULL`, this will cause a runtime error and your application will crash.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

audioburntest

HID Explorer

ImageClient

SeeMyFriends

TypeServicesForUnicode

Declared In

`CFBase.h`

CFRetain

Retains a Core Foundation object.

```
CFTypeRef CFRetain (
    CFTypeRef cf
);
```

Parameters

cf

>   The CFType object to retain. This value must not be `NULL`

Return Value

The input value, *cf*.

Discussion

You should retain a Core Foundation object when you receive it from elsewhere (that is, you did not create or copy it) and you want it to persist. If you retain a Core Foundation object you are responsible for releasing it (see *Memory Management Programming Guide for Core Foundation*).

Special Considerations

If *cf* is `NULL`, this will cause a runtime error and your application will crash.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

bulkerase

databurntest

MoreIsBetter

MoreSCF

QISA

Declared In

`CFBase.h`

CFShow

Prints a description of a Core Foundation object to stderr.

```
void CFShow (
    CFTypeRef obj
);
```

Parameters

obj

>   A Core Foundation object derived from CFType. If `obj` is not a Core Foundation object, an assertion is raised.

Discussion

The output is printed to the standard I/O standard error (stderr).

This function is useful as a debugging aid for Core Foundation objects. Because these objects are based on opaque types, it is difficult to examine their contents directly. However, the opaque types implement `description` function callbacks that return descriptions of their objects. This function invokes these callbacks.

Special Considerations

You can use `CFShow` in one of two general ways. If your debugger supports function calls (such as `gdb` does), call `CFShow` in the debugger:

```
(gdb) call (void) CFShow(string)
Hello World
```

You can also incorporate calls to `CFShow` in a test version of your code to print out "snapshots" of Core Foundation objects to the console.

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

Related Sample Code
FSFileOperation
MoreIsBetter
MoreSCF
ProfileSystem
QISA

Declared In
`CFString.h`

Data Types

CFHashCode

A type for hash codes returned by the `CFHash` function.

```
typedef unsigned long CFHashCode;
```

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

Declared In
`CFBase.h`

CFTypeID

A type for unique, constant integer values that identify particular Core Foundation opaque types.

```
typedef unsigned long CFTypeID;
```

Discussion
Defines a type identifier in Core Foundation. A type ID is an integer that identifies the opaque type to which a Core Foundation object "belongs." You use type IDs in various contexts, such as when you are operating on heterogeneous collections. Core Foundation provides programmatic interfaces for obtaining and evaluating type IDs.

Because the value for a type ID can change from release to release, your code should not rely on stored or hard-coded type IDs nor should it hard-code any observed properties of a type ID (such as, for example, it being a small integer).

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

Declared In
CFBase.h

CFTypeRef

An untyped "generic" reference to any Core Foundation object.

```
typedef const void * CFTypeRef;
```

Discussion
The CFTypeRef type is the base type defined in Core Foundation. It is used as the type and return value in several polymorphic functions. It is a generic object reference that acts as a placeholder for other true Core Foundation objects.

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

Declared In
CFBase.h

Document Revision History

This table describes the changes to *CFType Reference*.

Date	Notes
2007-07-12	Updated for Mac OS X v10.5.
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-03-03	TBD
2003-08-01	Corrected CFTypeRef declaration.
2003-01-01	First version of this document.

Index

C