
CType Reference

Data Management: Data Types & Collections



2007-07-12



Apple Inc.
© 2003, 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR

CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

CType Reference 5

Overview	5
Functions by Task	5
Memory Management	5
Determining Equality	5
Hashing	6
Miscellaneous Functions	6
Functions	6
CFCopyDescription	6
CFCopyTypeIDDescription	7
CFEqual	7
CFGetAllocator	8
CFGetRetainCount	8
CFGetTypeID	9
CFHash	10
CFMakeCollectable	10
CFRelease	11
CFRetain	11
CFShow	12
Data Types	13
CFHashCode	13
CTypeID	13
CTypeRef	14

Document Revision History 15

Index 17

CType 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 CType. The functions, callbacks, data types, and constants defined for CType can be used by any derived opaque type. Hence, CType functions are referred to as “polymorphic functions.” You use CType 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 (
    CTypeRef cf
);
```

Parameters

cf

The CType object (a generic reference of type [CTypeRef](#) (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

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 (
    CTypeID type_id
);
```

Parameters*theType*

An integer of type [CTypeID](#) (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

MoreSCF

Declared In

CFBase.h

CFEqual

Determines whether two Core Foundation objects are considered equal.

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

Parameters*cf1*

A CType object to compare to *cf2*.

cf2

A CType 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

audioburntest

bulkerase

databurntest

MoreSCF

Spotlight

Declared In

CFBase.h

CFGetAllocator

Returns the allocator used to allocate a Core Foundation object.

```
CFAllocatorRef CFGetAllocator (
    CTypeRef cf
);
```

Parameters

cf

The CType 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 (
    CTypeRef cf
);
```

Parameters

cf
The CType 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
MoreSCF

Declared In

CFBase.h

CFGetTypeID

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

```
CTypeID CFGetTypeID (
    CTypeRef cf
);
```

Parameters

cf
The CType object to examine.

Return Value

A value of type [CTypeID](#) (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 [CTypeID](#) (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

HID Calibrator
HID Config Save

HID Dumper
 HID Utilities
 MoreSCF

Declared In
 CFBase.h

CFHash

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

```
CFHashCode CFHash (
    CTypeRef cf
);
```

Parameters

cf
 A CType 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.

```
CTypeRef CFMakeCollectable (
    CTypeRef cf
);
```

Parameters

cf
 A CType 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

CIRAWFilterSample

Declared In

CFBase.h

CFRelease

Releases a Core Foundation object.

```
void CFRelease (  
    CTypeRef cf  
);
```

Parameters

cf

A CType 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

HID Calibrator

HID Config Save

HID Dumper

HID Utilities

ImageClient

Declared In

CFBase.h

CFRetain

Retains a Core Foundation object.

```
CTypeRef CFRetain (
    CTypeRef cf
);
```

Parameters*cf*

The CType 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

audioburntest

bulkerase

databurntest

MoreSCF

SampleHardwarePlugIn

Declared In

CFBase.h

CFShow

Prints a description of a Core Foundation object to stderr.

```
void CFShow (
    CTypeRef obj
);
```

Parameters*obj*

A Core Foundation object derived from CType. 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

HID Utilities Source

MoreSCF

ProfileSystem

USBPrivateDataSample

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

CTypeID

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

```
typedef unsigned long CTypeID;
```

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

CTypeRef

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

```
typedef const void * CTypeRef;
```

Discussion

The `CTypeRef` 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 *CType 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 CTypeRef declaration.
2003-01-01	First version of this document.

REVISION HISTORY

Document Revision History

Index

C

CFCopyDescription **function** [6](#)
CFCopyTypeIDDescription **function** [7](#)
CFEqual **function** [7](#)
CFGetAllocator **function** [8](#)
CFGetRetainCount **function** [8](#)
CFGetTypeID **function** [9](#)
CFHash **function** [10](#)
CFHashCode **data type** [13](#)
CFMakeCollectable **function** [10](#)
CFRelease **function** [11](#)
CFRetain **function** [11](#)
CFShow **function** [12](#)
CFTypeID **data type** [13](#)
CFTypeRef **data type** [14](#)