
CFNumber Reference

Data Management: Dates, Times, & Numbers



2006-12-01



Apple Inc.
© 2003, 2006 Apple Computer, 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, Carbon, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

iPhone and Numbers are trademarks of Apple Inc.

Times is a registered trademark of Heidelberger Druckmaschinen AG, available from Linotype Library GmbH.

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

CFNumber Reference 5

Overview	5
Functions by Task	5
Creating a Number	5
Getting Information About Numbers	6
Comparing Numbers	6
Getting the CFNumber Type ID	6
Functions	6
CFNumberCompare	6
CFNumberCreate	7
CFNumberGetByteSize	8
CFNumberGetType	8
CFNumberGetTypeID	9
CFNumberGetValue	9
CFNumberIsFloatType	10
Data Types	11
CFNumberRef	11
Constants	11
Number Types	11
Predefined Values	13

Document Revision History 15

Index 17

CFNumber Reference

Derived From:	CFPropertyList : CType
Framework:	CoreFoundation/CoreFoundation.h
Companion guide	Property List Programming Topics for Core Foundation
Declared in	CFNumber.h

Overview

CFNumber encapsulates C scalar (numeric) types. It provides functions for setting and accessing the value as any basic C type. It also provides a compare function to determine the ordering of two CFNumber objects. CFNumber objects are used to wrap numerical values for use in Core Foundation property lists and collections.

CFNumber objects are not intended as a replacement for C scalar values and should not be used in APIs or implementations where scalar values are more appropriate and efficient.

Note: In order to improve performance, some commonly-used numbers (such as 0 and 1) are uniqued. You should not expect that allocating multiple CFNumber instances will necessarily result in distinct objects.

CFNumber is “toll-free bridged” with its Cocoa Foundation counterpart, NSNumber. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSNumber *` parameter, you can pass in a `CFNumberRef`, and in a function where you see a `CFNumberRef` parameter, you can pass in an `NSNumber` instance. This fact also applies to concrete subclasses of `NSNumber`. See *Integrating Carbon and Cocoa in Your Application* for more information on toll-free bridging.

Functions by Task

Creating a Number

[CFNumberCreate](#) (page 7)

Creates a CFNumber object using a specified value.

Getting Information About Numbers

[CFNumberGetByteSize](#) (page 8)

Returns the number of bytes used by a CFNumber object to store its value.

[CFNumberGetType](#) (page 8)

Returns the type used by a CFNumber object to store its value.

[CFNumberGetValue](#) (page 9)

Obtains the value of a CFNumber object cast to a specified type.

[CFNumberIsFloatType](#) (page 10)

Determines whether a CFNumber object contains a value stored as one of the defined floating point types.

Comparing Numbers

[CFNumberCompare](#) (page 6)

Compares two CFNumber objects and returns a comparison result.

Getting the CFNumber Type ID

[CFNumberGetTypeID](#) (page 9)

Returns the type identifier for the CFNumber opaque type.

Functions

CFNumberCompare

Compares two CFNumber objects and returns a comparison result.

```
CFComparisonResult CFNumberCompare (
    CFNumberRef number,
    CFNumberRef otherNumber,
    void *context
);
```

Parameters

number

The first CFNumber object to compare.

otherNumber

The second CFNumber object to compare.

context

Pass NULL.

Return Value

A `CFComparisonResult` constant that indicates whether *number* is equal to, less than, or greater than *otherNumber*.

Discussion

When comparing two CFNumber objects using this function, one or both objects can represent a special-case number such as signed 0, signed infinity, or NaN.

The following rules apply:

- Negative 0 compares less than positive 0.
- Positive infinity compares greater than everything except itself, to which it compares equal.
- Negative infinity compares less than everything except itself, to which it compares equal.
- If both numbers are NaN, then they compare equal.
- If only one of the numbers is NaN, then the NaN compares greater than the other number if it is negative, and smaller than the other number if it is positive.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFNumber.h

CFNumberCreate

Creates a CFNumber object using a specified value.

```
CFNumberRef CFNumberCreate (
    CFAAllocatorRef allocator,
    CFNumberType theType,
    const void *valuePtr
);
```

Parameters

allocator

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

theType

A constant that specifies the data type of the value to convert. See [Number Types](#) (page 11) for a list of possible values.

valuePtr

A pointer to the value for the returned number object.

Return Value

A new number with the value specified by *valuePtr*. Ownership follows the Create Rule.

Discussion

The *theType* parameter is not necessarily preserved when creating a new CFNumber object. The CFNumber object will be created using whatever internal storage type the creation function deems appropriate. Use the function [CFNumberGetType](#) (page 8) to find out what type the CFNumber object used to store your value.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

audioburntest

BetterAuthorizationSample

BSDLLCTest

HID Explorer

QTMetaData

Declared In

CFNumber.h

CFNumberGetByteSize

Returns the number of bytes used by a CFNumber object to store its value.

```
CFIndex CFNumberGetByteSize (
    CFNumberRef number
);
```

Parameters

number

The CFNumber object to examine.

Return Value

The size in bytes of the value contained in *number*.

Discussion

Because a CFNumber object might store a value using a type different from that of the original value with which it was created, this function may return a size different from the size of the original value's type.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFNumber.h

CFNumberGetType

Returns the type used by a CFNumber object to store its value.

```
CFNumberType CFNumberGetType (
    CFNumberRef number
);
```

Parameters

number

The CFNumber object to examine.

Return Value

A constant that indicates the data type of the value contained in *number*. See [Number Types](#) (page 11) for a list of possible values.

Discussion

The type specified in the call to [CFNumberCreate](#) (page 7) is not necessarily preserved when a new CFNumber object is created—it uses whatever internal storage type the creation function deems appropriate.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFNumber.h

CFNumberGetTypeID

Returns the type identifier for the CFNumber opaque type.

```
CFTypeID CFNumberGetTypeID (
    void
);
```

Return Value

The type identifier for the CFNumber opaque type.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

BetterAuthorizationSample

BSDLLCTest

CFFTPSample

HID Manager Basics

MoreSCF

Declared In

CFNumber.h

CFNumberGetValue

Obtains the value of a CFNumber object cast to a specified type.

```
Boolean CFNumberGetValue (
    CFNumberRef number,
    CFNumberType theType,
    void *valuePtr
);
```

Parameters

number

The CFNumber object to examine.

theType

A constant that specifies the data type to return. See [Number Types](#) (page 11) for a list of possible values.

valuePtr

On return, contains the value of *number*.

Return Value

`true` if the operation was successful, otherwise `false`.

Discussion

If the argument type differs from the return type, and the conversion is lossy or the return value is out of range, then this function passes back an approximate value in *valuePtr* and returns `false`.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

AudioCDSample

BSDLLCTest

GLUT

HID Manager Basics

MoreSCF

Declared In

CFNumber.h

CFNumberIsFloatType

Determines whether a CFNumber object contains a value stored as one of the defined floating point types.

```
Boolean CFNumberIsFloatType (
    CFNumberRef number
);
```

Parameters

number

The CFNumber object to examine.

Return Value

`true` if *number*'s value is one of the defined floating point types, otherwise `false`. The valid floating point types are listed in [Number Types](#) (page 11).

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFNumber.h

Data Types

CFNumberRef

A reference to a CFNumber object.

```
typedef const struct __CFNumber *CFNumberRef;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

CFNumber.h

Constants

Number Types

Flags used by CFNumber to indicate the data type of a value.

```
enum CFNumberType {
    kCFNumberSInt8Type = 1,
    kCFNumberSInt16Type = 2,
    kCFNumberSInt32Type = 3,
    kCFNumberSInt64Type = 4,
    kCFNumberFloat32Type = 5,
    kCFNumberFloat64Type = 6,
    kCFNumberCharType = 7,
    kCFNumberShortType = 8,
    kCFNumberIntType = 9,
    kCFNumberLongType = 10,
    kCFNumberLongLongType = 11,
    kCFNumberFloatType = 12,
    kCFNumberDoubleType = 13,
    kCFNumberCFIndexType = 14,
    kCFNumberNSIntegerType = 15,
    kCFNumberCGFloatType = 16,
    kCFNumberMaxType = 16
};
typedef enum CFNumberType CFNumberType;
```

Constants

kCFNumberSInt8Type

Eight-bit, signed integer. The SInt8 data type is defined in MacTypes.h.

Available in Mac OS X v10.0 and later.

Declared in CFNumber.h.

`kCFNumberSInt16Type`

Sixteen-bit, signed integer. The `SInt16` data type is defined in `MacTypes.h`.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberSInt32Type`

Thirty-two-bit, signed integer. The `SInt32` data type is defined in `MacTypes.h`.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberSInt64Type`

Sixty-four-bit, signed integer. The `SInt64` data type is defined in `MacTypes.h`.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberFloat32Type`

Thirty-two-bit real. The `Float32` data type is defined in `MacTypes.h`.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberFloat64Type`

Sixty-four-bit real. The `Float64` data type is defined in `MacTypes.h` and conforms to the 64-bit IEEE 754 standard.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberCharType`

Basic C `char` type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberShortType`

Basic C `short` type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberIntType`

Basic C `int` type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberLongType`

Basic C `long` type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberLongLongType`

Basic C `long long` type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberFloatType`

Basic C float type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberDoubleType`

Basic C double type.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberCFIndexType`

CFIndex value.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberNSIntegerType`

NSInteger value.

Available in Mac OS X v10.5 and later.

Declared in `CFNumber.h`.

`kCFNumberCGFloatType`

CGFloat value.

Available in Mac OS X v10.5 and later.

Declared in `CFNumber.h`.

`kCFNumberMaxType`

Same as `kCFNumberCGFloatType`.

Note that on Mac OS X v10.4, `kCFNumberMaxType` was the same as `kCFNumberCFIndexType`.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

Discussion

The type specified in the call to `CFNumberCreate` (page 7) is not necessarily preserved when creating a new `CFNumber` object. A `CFNumber` object uses whatever internal storage type the creation function deems appropriate. Use the `CFNumberGetType` (page 8) function to find out what type the `CFNumber` object used to store your value.

Predefined Values

`CFNumber` provides some predefined number values.

```
const CFNumberRef kCFNumberNaN;
const CFNumberRef kCFNumberNegativeInfinity;
const CFNumberRef kCFNumberPositiveInfinity;
```

Constants

`kCFNumberNaN`

“Not a Number.” This value is often the result of an invalid operation, such as the square-root of a negative number.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberNegativeInfinity`

Designates a negative infinity value.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

`kCFNumberPositiveInfinity`

Designates a positive infinity value.

Available in Mac OS X v10.0 and later.

Declared in `CFNumber.h`.

Document Revision History

This table describes the changes to *CFNumber Reference*.

Date	Notes
2006-12-01	Updated to include new API in Mac OS X v10.5.
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-04-29	Added note to Introduction regarding uniquing of commonly-used numbers.
2003-01-01	First version of this document.

REVISION HISTORY

Document Revision History

Index

C

CFNumberCompare **function** [6](#)
CFNumberCreate **function** [7](#)
CFNumberGetByteSize **function** [8](#)
CFNumberGetType **function** [8](#)
CFNumberGetTypeID **function** [9](#)
CFNumberGetValue **function** [9](#)
CFNumberIsFloatType **function** [10](#)
CFNumberRef **data type** [11](#)

K

kCFNumberCFIndexType **constant** [13](#)
kCFNumberCGFloatType **constant** [13](#)
kCFNumberCharType **constant** [12](#)
kCFNumberDoubleType **constant** [13](#)
kCFNumberFloat32Type **constant** [12](#)
kCFNumberFloat64Type **constant** [12](#)
kCFNumberFloatType **constant** [13](#)
kCFNumberIntType **constant** [12](#)
kCFNumberLongLongType **constant** [12](#)
kCFNumberLongType **constant** [12](#)
kCFNumberMaxType **constant** [13](#)
kCFNumberNaN **constant** [13](#)
kCFNumberNegativeInfinity **constant** [14](#)
kCFNumberNSIntegerType **constant** [13](#)
kCFNumberPositiveInfinity **constant** [14](#)
kCFNumberShortType **constant** [12](#)
kCFNumberSInt16Type **constant** [12](#)
kCFNumberSInt32Type **constant** [12](#)
kCFNumberSInt64Type **constant** [12](#)
kCFNumberSInt8Type **constant** [11](#)

N

Number Types [11](#)

P

Predefined Values [13](#)