
CFUUID Reference

Core Foundation



2007-01-15



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 and the Apple logo 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

CFUUID Reference 5

- Overview 5
- Functions by Task 5
 - Creating CFUUID Objects 5
 - Getting Information About CFUUID Objects 6
 - Getting the CFUUID Type Identifier 6
- Functions 6
 - CFUUIDCreate 6
 - CFUUIDCreateFromString 7
 - CFUUIDCreateFromUUIDBytes 7
 - CFUUIDCreateString 8
 - CFUUIDCreateWithBytes 9
 - CFUUIDGetConstantUUIDWithBytes 10
 - CFUUIDGetTypeID 12
 - CFUUIDGetUUIDBytes 12
- Data Types 12
 - CFUUIDBytes 12
 - CFUUIDRef 14

Document Revision History 15

Index 17

CFUUID Reference

Derived From:	CType
Framework:	CoreFoundation/CoreFoundation.h
Companion guide	Plug-ins
Declared in	CFUUID.h

Overview

CFUUID objects are used by plug-ins to uniquely identify types, interfaces, and factories. When creating a new type, host developers must generate UUIDs to identify the type as well as its interfaces and factories.

UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers) or IIDs (Interface Identifiers), are 128-bit values guaranteed to be unique. A UUID is made unique over both space and time by combining a value unique to the computer on which it was generated—usually the Ethernet hardware address—and a value representing the number of 100-nanosecond intervals since October 15, 1582 at 00:00:00.

The standard format for UUIDs represented in ASCII is a string punctuated by hyphens, for example 68753A44-4D6F-1226-9C60-0050E4C00067. The hex representation looks, as you might expect, like a list of numerical values preceded by 0x. For example, 0xD7, 0x36, 0x95, 0x0A, 0x4D, 0x6E, 0x12, 0x26, 0x80, 0x3A, 0x00, 0x50, 0xE4, 0xC0, 0x00, 0x67. To use a UUID, you simply create it and then copy the resulting strings into your header and C language source files. Because a UUID is expressed simply as an array of bytes, there are no endianness considerations for different platforms.

You can create a CFUUID object, and thereby generate a UUID, using any one of the `CFUUIDCreate...` functions. Use the [CFUUIDGetConstantUUIDWithBytes](#) (page 10) function if you want to declare a UUID constant in a `#define` statement. You can get the raw bytes of an existing CFUUID object using the [CFUUIDGetUUIDBytes](#) (page 12) function.

Functions by Task

Creating CFUUID Objects

[CFUUIDCreate](#) (page 6)

Creates a Universally Unique Identifier (UUID) object.

[CFUUIDCreateFromString](#) (page 7)

Creates a CFUUID object for a specified string.

[CFUUIDCreateFromUUIDBytes](#) (page 7)

Creates a CFUUID object from raw UUID bytes.

[CFUUIDCreateWithBytes](#) (page 9)

Creates a CFUUID object from raw UUID bytes.

Getting Information About CFUUID Objects

[CFUUIDCreateString](#) (page 8)

Returns the string representation of a specified CFUUID object.

[CFUUIDGetConstantUUIDWithBytes](#) (page 10)

Returns a CFUUID object from raw UUID bytes.

[CFUUIDGetUUIDBytes](#) (page 12)

Returns the value of a UUID object as raw bytes.

Getting the CFUUID Type Identifier

[CFUUIDGetTypeID](#) (page 12)

Returns the type identifier for all CFUUID objects.

Functions

CFUUIDCreate

Creates a Universally Unique Identifier (UUID) object.

```
CFUUIDRef CFUUIDCreate (
    CFAllocatorRef alloc
);
```

Parameters

alloc

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

Return Value

A new CFUUID object. Ownership follows the Create Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

AutomatorHandsOn

People

StickiesExample

Declared In
CFUUID.h

CFUUIDCreateFromString

Creates a CFUUID object for a specified string.

```
CFUUIDRef CFUUIDCreateFromString (
    CFAllocatorRef alloc,
    CFStringRef uuidStr
);
```

Parameters

alloc

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

uuidStr

A string containing a UUID. The standard format for UUIDs represented in ASCII is a string punctuated by hyphens, for example `68753A44-4D6F-1226-9C60-0050E4C00067`.

Return Value

A new CFUUID object, or if a CFUUID object of the same value already exists, the existing instance with its reference count incremented. Ownership follows the Create Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

CoreRecipes

Departments and Employees

PDEProject

Spotlight

SpotlightFortunes

Declared In
CFUUID.h

CFUUIDCreateFromUUIDBytes

Creates a CFUUID object from raw UUID bytes.

```
CFUUIDRef CFUUIDCreateFromUUIDBytes (
    CFAllocatorRef alloc,
    CFUUIDBytes bytes
);
```

Parameters

alloc

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

bytes

Raw UUID bytes to use to create the CFUUID object.

Return Value

A new CFUUID object. Ownership follows the Create Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

BasicPlugIn

CoreRecipes

Departments and Employees

Spotlight

SpotlightFortunes

Declared In

CFUUID.h

CFUUIDCreateString

Returns the string representation of a specified CFUUID object.

```
CFStringRef CFUUIDCreateString (
    CFAllocatorRef alloc,
    CFUUIDRef uuid
);
```

Parameters

alloc

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

uuid

The CFUUID object whose string representation to obtain.

Return Value

The string representation of *uuid*. Ownership follows the Create Rule.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

AutomatorHandsOn

IdentitySample

PDEProject

People

StickiesExample

Declared In

CFUUID.h

CFUUIDCreateWithBytes

Creates a CFUUID object from raw UUID bytes.

```
CFUUIDRef CFUUIDCreateWithBytes (
    CFAAllocatorRef alloc,
    UInt8 byte0,
    UInt8 byte1,
    UInt8 byte2,
    UInt8 byte3,
    UInt8 byte4,
    UInt8 byte5,
    UInt8 byte6,
    UInt8 byte7,
    UInt8 byte8,
    UInt8 byte9,
    UInt8 byte10,
    UInt8 byte11,
    UInt8 byte12,
    UInt8 byte13,
    UInt8 byte14,
    UInt8 byte15
);
```

Parameters

alloc

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

byte0

Raw byte number 0.

byte1

Raw byte number 1.

byte2

Raw byte number 2.

byte3

Raw byte number 3.

byte4

Raw byte number 4.

byte5

Raw byte number 5.

byte6

Raw byte number 6.

byte7

Raw byte number 7.

byte8

Raw byte number 8.

byte9

Raw byte number 9.

byte10

Raw byte number 10.

byte11

Raw byte number 11.

byte12

Raw byte number 12.

byte13

Raw byte number 13.

byte14

Raw byte number 14.

byte15

Raw byte number 15.

Return Value

A new CFUUID object, or, if a CFUUID object of the same value already exists, the existing instance with its reference count incremented. Ownership follows the Create Rule.

Discussion

.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFUUID.h

CFUUIDGetConstantUUIDWithBytes

Returns a CFUUID object from raw UUID bytes.

```
CFUUIDRef CFUUIDGetConstantUUIDWithBytes (
    CFAllocatorRef alloc,
    UInt8 byte0,
    UInt8 byte1,
    UInt8 byte2,
    UInt8 byte3,
    UInt8 byte4,
    UInt8 byte5,
    UInt8 byte6,
    UInt8 byte7,
    UInt8 byte8,
    UInt8 byte9,
    UInt8 byte10,
    UInt8 byte11,
    UInt8 byte12,
    UInt8 byte13,
    UInt8 byte14,
    UInt8 byte15
);
```

Parameters*alloc*

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

- byte0*
Raw byte number 0.
- byte1*
Raw byte number 1.
- byte2*
Raw byte number 2.
- byte3*
Raw byte number 3.
- byte4*
Raw byte number 4.
- byte5*
Raw byte number 5.
- byte6*
Raw byte number 6.
- byte7*
Raw byte number 7.
- byte8*
Raw byte number 8.
- byte9*
Raw byte number 9.
- byte10*
Raw byte number 10.
- byte11*
Raw byte number 11.
- byte12*
Raw byte number 12.
- byte13*
Raw byte number 13.
- byte14*
Raw byte number 14.
- byte15*
Raw byte number 15.

Return Value

A CFUUID object. Ownership follows the Get Rule.

Discussion

This function can be used in headers to declare a UUID constant with `#define`.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFUUID.h

CFUUIDGetTypeID

Returns the type identifier for all CFUUID objects.

```
CTypeID CFUUIDGetTypeID (
    void
);
```

Return Value

The type identifier for the CFUUID opaque type.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFUUID.h

CFUUIDGetUUIDBytes

Returns the value of a UUID object as raw bytes.

```
CFUUIDBytes CFUUIDGetUUIDBytes (
    CFUUIDRef uuid
);
```

Parameters

uuid

The CFUUID object to examine.

Return Value

The value of *uuid* represented as raw bytes.

Availability

Available in CarbonLib v1.1 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code

BasicPlugIn

SCSIOldAndNew

simpleAVC

USBPrivateDataSample

Declared In

CFUUID.h

Data Types

CFUUIDBytes

A 128-bit struct that represents a UUID as raw bytes.

```
typedef struct {
    UInt8 byte0;
    UInt8 byte1;
    UInt8 byte2;
    UInt8 byte3;
    UInt8 byte4;
    UInt8 byte5;
    UInt8 byte6;
    UInt8 byte7;
    UInt8 byte8;
    UInt8 byte9;
    UInt8 byte10;
    UInt8 byte11;
    UInt8 byte12;
    UInt8 byte13;
    UInt8 byte14;
    UInt8 byte15;
} CFUUIDBytes;
```

Fields

byte0

The first byte.

byte1

The second byte.

byte2

The third byte.

byte3

The fourth byte.

byte4

The fifth byte.

byte5

The sixth byte.

byte6

The seventh byte.

byte7

The eighth byte.

byte8

The ninth byte.

byte9

The tenth byte.

byte10

The eleventh byte.

byte11

The twelfth byte.

byte12

The thirteenth byte.

byte13

The fourteenth byte.

byte14

The fifteenth byte.

byte15

The sixteenth byte.

Discussion

This structure can be obtained from a CFUUID object using the [CFUUIDGetUUIDBytes](#) (page 12) function. This structure is can be passed to functions that expect a raw UUID.

Availability

Available in Mac OS X v10.0 and later.

Declared In

CFUUID.h

CFUUIDRef

A reference to a CFUUID object.

```
typedef const struct __CFUUID *CFUUIDRef;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

CFUUID.h

Document Revision History

This table describes the changes to *CFUUID Reference*.

Date	Notes
2007-01-15	Corrected minor typographical error.
2006-04-04	Added assurance that there are no endian issues.
2005-12-06	Made minor changes to conform to documentation guidelines.
2005-08-11	Cosmetic changes to conform to documentation guidelines.
2003-01-01	First version of this document.

REVISION HISTORY

Document Revision History

Index

C

CFUUIDBytes **structure** [12](#)
CFUUIDCreate **function** [6](#)
CFUUIDCreateFromString **function** [7](#)
CFUUIDCreateFromUUIDBytes **function** [7](#)
CFUUIDCreateString **function** [8](#)
CFUUIDCreateWithBytes **function** [9](#)
CFUUIDGetConstantUUIDWithBytes **function** [10](#)
CFUUIDGetTypeID **function** [12](#)
CFUUIDGetUUIDBytes **function** [12](#)
CFUUIDRef **data type** [14](#)