

---

# CFMutableData Reference

Core Foundation



2007-03-07



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, Carbon, and Cocoa 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

---

## **CFMutableData Reference 5**

---

Overview	5
Functions	5
CFDataAppendBytes	5
CFDataCreateMutable	6
CFDataCreateMutableCopy	7
CFDataDeleteBytes	7
CFDataGetMutableBytePtr	8
CFDataIncreaseLength	8
CFDataReplaceBytes	9
CFDataSetLength	9
Data Types	10
CFMutableDataRef	10

---

## **Document Revision History 11**

---

## **Index 13**

---



# CFMutableData Reference

---

<b>Derived From:</b>	CFData : CFPropertyList : CType
<b>Framework:</b>	CoreFoundation/CoreFoundation.h
<b>Declared in</b>	CFData.h
<b>Companion guides</b>	Binary Data Programming Guide for Core Foundation Property List Programming Topics for Core Foundation

## Overview

CFMutableData manages dynamic binary data. The basic interface for managing binary data is provided by CFData. CFMutableData adds functions to modify the contents of a binary data object.

You create a mutable data object using either the [CFDataCreateMutable](#) (page 6) or [CFDataCreateMutableCopy](#) (page 7) function.

Bytes are added to a data object with the [CFDataAppendBytes](#) (page 5) function. Bytes are removed from a data object with the [CFDataDeleteBytes](#) (page 7) function.

CFMutableData is “toll-free bridged” with its Cocoa Foundation counterpart, NSMutableData. What this means is that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. In other words, in a method where you see an `NSMutableData *` parameter, you can pass in a `CFMutableDataRef`, and in a function where you see a `CFMutableDataRef` parameter, you can pass in an `NSMutableData` instance. This also applies to concrete subclasses of `NSMutableData`. See [Interchangeable Data Types](#) for more information on toll-free bridging.

## Functions

### CFDataAppendBytes

Appends the bytes from a byte buffer to the contents of a CFData object.

```
void CFDataAppendBytes (
    CFMutableDataRef theData,
    const UInt8 *bytes,
    CFIndex length
);
```

**Parameters***theData*

A CFMutableData object. If you pass an immutable CFData object, the behavior is not defined.

*bytes*

A pointer to the buffer of bytes to be added to *theData*.

*length*

The number of bytes in the byte buffer *bytes*.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

CarbonSketch

CFLocalServer

ImageClient

**Declared In**

CFData.h

**CFDataCreateMutable**

Creates an empty CFMutableData object.

```
CFMutableDataRef CFDataCreateMutable (
    CFAllocatorRef allocator,
    CFIndex capacity
);
```

**Parameters***allocator*

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

*capacity*

The maximum number of bytes that the CFData object can contain. If 0, the object can grow to a size only limited by the constraints of available memory and address space.

**Return Value**

A CFMutableData object or NULL if there was a problem creating the object. Ownership follows the Create Rule.

**Discussion**

This function creates an empty (that is, content-less) CFMutableData object. You can add raw data to this object with the [CFDataAppendBytes](#) (page 5) function, and thereafter you can replace and delete characters with the appropriate CFMutableData functions. If the *capacity* parameter is greater than 0, any attempt to add characters beyond this limit can result in undefined behavior.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

CarbonSketch

CFLocalServer

ImageClient

SeeMyFriends

UnsharpMask

**Declared In**

CFData.h

**CFDataCreateMutableCopy**

Creates a CFMutableData object by copying another CFData object.

```
CFMutableDataRef CFDataCreateMutableCopy (
    CFAllocatorRef allocator,
    CFIndex capacity,
    CFDataRef theData
);
```

**Parameters**

*allocator*

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

*capacity*

The maximum number of bytes the object should contain. If 0, the object can grow to a size only limited by the constraints of available memory and address space. Note that initially the created CFData object still has the same length as the original object; this parameter simply specifies what the maximum size is. CFData might try to optimize its internal storage by paying attention to this value.

*theData*

The CFData object to be copied.

**Return Value**

A CFMutableData object that has the same contents as the original object. Returns `NULL` if there was a problem copying the object. Ownership follows the Create Rule.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFData.h

**CFDataDeleteBytes**

Deletes the bytes in a CFMutableData object within a specified range.

```
void CFDataDeleteBytes (
    CFMutableDataRef theData,
    CFRange range
);
```

**Parameters***theData*

A CFMutableData object. If you pass an immutable CFData object, the behavior is not defined.

*range*

The range of bytes (that is, the starting byte and the number of bytes from that point) to delete from *theData*'s byte buffer.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

CFLocalServer

**Declared In**

CFData.h

**CFDataGetMutableBytePtr**

Returns a pointer to a mutable byte buffer of a CFMutableData object.

```
UInt8 *CFDataGetMutableBytePtr (
    CFMutableDataRef theData
);
```

**Parameters***theData*

A CFMutableData object. If you pass an immutable CFData object, the behavior is not defined.

**Return Value**

A pointer to the bytes associated with *theData*.

**Discussion**

If the length of *theData*'s data is not zero, this function is guaranteed to return a pointer to a CFMutableData object's internal bytes. If the length of *theData*'s data is zero, this function may or may not return NULL dependent upon many factors related to how the object was created (moreover, in this case the function result might change between different releases and on different platforms).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFData.h

**CFDataIncreaseLength**

Increases the length of a CFMutableData object's internal byte buffer, zero-filling the extension to the buffer.



```
void CFDataIncreaseLength (
    CFMutableDataRef theData,
    CFIndex extraLength
);
```

**Parameters***theData*

A CFMutableData object. If you pass an immutable CFData object, the behavior is not defined.

*extraLength*

The number of bytes by which to increase the byte buffer.

**Discussion**

This function increases the length of a CFMutableData object's underlying byte buffer to a new size, initializing the new bytes to 0.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFData.h

**CFDataReplaceBytes**

Replaces those bytes in a CFMutableData object that fall within a specified range with other bytes.

```
void CFDataReplaceBytes (
    CFMutableDataRef theData,
    CFRange range,
    const UInt8 *newBytes,
    CFIndex newLength
);
```

**Parameters***theData*

A CFMutableData object. If you pass an immutable CFData object, the behavior is not defined.

*range*

The range of bytes (that is, the starting byte and the number of bytes from that point) to delete from *theData's* byte buffer.

*newBytes*

A pointer to the buffer containing the replacement bytes.

*newLength*

The number of bytes in the byte buffer *newBytes*.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFData.h

**CFDataSetLength**

Resets the length of a CFMutableData object's internal byte buffer.

```
void CFDataSetLength (
    CFMutableDataRef theData,
    CFIndex length
);
```

**Parameters***theData*

A CFMutableData object. If you pass an immutable CFData object, the behavior is not defined.

*length*

The new size of *theData's* byte buffer.

**Discussion**

This function resets the length of a CFMutableData object's underlying byte buffer to a new size. If that size is less than the current size, it truncates the excess bytes. If that size is greater than the current size, it zero-fills the extension to the byte buffer.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFData.h

## Data Types

**CFMutableDataRef**

A reference to a CFMutableData object.

```
typedef struct __CFData *CFMutableDataRef;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFData.h

# Document Revision History

---

This table describes the changes to *CFMutableData Reference*.

Date	Notes
2007-03-07	Corrected definition of <code>CFDataGetMutableBytePtr</code> .
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-08-11	Clarified pointer validity after mutation in <code>CFDataGetMutableBytePtr</code> .
2003-08-01	Added link to Carbon-Cocoa integration document.
2003-01-01	First version of this document.

## REVISION HISTORY

### Document Revision History

# Index

---

## C

---

CFDataAppendBytes **function** [5](#)  
CFDataCreateMutable **function** [6](#)  
CFDataCreateMutableCopy **function** [7](#)  
CFDataDeleteBytes **function** [7](#)  
CFDataGetMutableBytePtr **function** [8](#)  
CFDataIncreaseLength **function** [8](#)  
CFDataReplaceBytes **function** [9](#)  
CFDataSetLength **function** [9](#)  
CFMutableDataRef **data type** [10](#)