

---

# NSMutableDictionary Reference

Data Management: Data Types & Collections





Apple Inc.  
© 2003, 2009 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, Cocoa, and iPhone are trademarks of Apple Inc., registered in the United States and other countries.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

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

---

## **NSMutableDictionary Reference 5**

---

- Overview 5
- Functions by Task 5
  - Creating a Mutable Dictionary 5
  - Modifying a Dictionary 6
- Functions 6
  - NSMutableDictionaryAddValue 6
  - NSMutableDictionaryCreateMutable 6
  - NSMutableDictionaryCreateMutableCopy 8
  - NSMutableDictionaryRemoveAllValues 8
  - NSMutableDictionaryRemoveValue 9
  - NSMutableDictionaryReplaceValue 9
  - NSMutableDictionarySetValue 9
- Data Types 10
  - NSMutableDictionaryRef 10

---

## **Document Revision History 11**

---



# NSMutableDictionary Reference

---

<b>Derived From:</b>	<i>CFDictionary Reference</i> : <i>CFPropertyList Reference</i> : <i>CFTYPE Reference</i>
<b>Framework:</b>	CoreFoundation/CoreFoundation.h
<b>Declared in</b>	CFDictionary.h
<b>Companion guides</b>	Collections Programming Topics for Core Foundation Property List Programming Topics for Core Foundation

## Overview

NSMutableDictionary manages dynamic dictionaries. The basic interface for managing dictionaries is provided by *CFDictionary Reference*. NSMutableDictionary adds functions to modify the contents of a dictionary.

You create a mutable dictionary object using either the [CFDictionaryCreateMutable](#) (page 6) or [CFDictionaryCreateMutableCopy](#) (page 8) function. You can add key-value pairs using the [CFDictionaryAddValue](#) (page 6) and [CFDictionarySetValue](#) (page 9) functions. When adding key-value pairs to a dictionary, the keys and values are not copied—they are retained so they are not invalidated before the dictionary is deallocated. You can remove key-value pairs using the [CFDictionaryRemoveValue](#) (page 9) function. When removing key-value pairs from a dictionary, the keys and values are released.

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

## Functions by Task

### Creating a Mutable Dictionary

[CFDictionaryCreateMutable](#) (page 6)  
Creates a new mutable dictionary.

[CFDictionaryCreateMutableCopy](#) (page 8)  
Creates a new mutable dictionary with the key-value pairs from another dictionary.

## Modifying a Dictionary

[CFDictionaryAddValue](#) (page 6)

Adds a key-value pair to a dictionary if the specified key is not already present.

[CFDictionaryRemoveAllValues](#) (page 8)

Removes all the key-value pairs from a dictionary, making it empty.

[CFDictionaryRemoveValue](#) (page 9)

Removes a key-value pair.

[CFDictionaryReplaceValue](#) (page 9)

Replaces a value corresponding to a given key.

[CFDictionarySetValue](#) (page 9)

Sets the value corresponding to a given key.

## Functions

### CFDictionaryAddValue

Adds a key-value pair to a dictionary if the specified key is not already present.

```
void CFDictionaryAddValue (
    CFMutableDictionaryRef theDict,
    const void *key,
    const void *value
);
```

#### Parameters

*theDict*

The dictionary to modify. If the dictionary is a fixed-capacity dictionary and it is full before this operation, the behavior is undefined.

*key*

The key for the value to add to the dictionary—a CType object or a pointer value. The *key* is retained by the dictionary using the retain callback provided when the dictionary was created, so must be of the type expected by the callback. If a key which matches *key* is already present in the dictionary, this function does nothing ("add if absent").

*value*

A CType object or a pointer value to add to the dictionary. The *value* is retained by the dictionary using the retain callback provided when the dictionary was created, so must be of the type expected by the callback.

#### Availability

Available in iOS 2.0 and later.

#### Declared In

CFDictionary.h

### CFDictionaryCreateMutable

Creates a new mutable dictionary.

```
CFMutableDictionaryRef CFDictionaryCreateMutable (
    CFAllocatorRef allocator,
    CFIndex capacity,
    const CFDictionaryKeyCallBacks *keyCallBacks,
    const CFDictionaryValueCallBacks *valueCallBacks
);
```

**Parameters***allocator*

The allocator to use to allocate memory for the new dictionary and its storage for key-value pairs. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

*capacity*

The maximum number of key-value pairs that can be contained by the new dictionary. The dictionary starts empty and can grow to this number of key-value pairs (and it can have less).

Pass 0 to specify that the maximum capacity is not limited. The value must not be negative.

*keyCallBacks*

A pointer to a `CFDictionaryKeyCallBacks` structure initialized with the callbacks to use to retain, release, describe, and compare keys in the dictionary. A copy of the contents of the callbacks structure is made, so that a pointer to a structure on the stack can be passed in or can be reused for multiple collection creations.

This value may be `NULL`, which is treated as a valid structure of version 0 with all fields `NULL`. Otherwise, if any of the fields are not valid pointers to functions of the correct type, or this value is not a valid pointer to a `CFDictionaryKeyCallBacks` structure, the behavior is undefined. If any of the keys put into the collection is not one understood by one of the callback functions, the behavior when that callback function is used is undefined.

If the dictionary will contain only `CType` objects, then pass a pointer to `kCTypeDictionaryKeyCallBacks` as this parameter to use the default callback functions.

*valueCallBacks*

A pointer to a `CFDictionaryValueCallBacks` structure initialized with the callbacks to use to retain, release, describe, and compare values in the dictionary. A copy of the contents of the callbacks structure is made, so that a pointer to a structure on the stack can be passed in or can be reused for multiple collection creations.

This value may be `NULL`, which is treated as a valid structure of version 0 with all fields `NULL`. Otherwise, if any of the fields are not valid pointers to functions of the correct type, or this value is not a valid pointer to a `CFDictionaryValueCallBacks` structure, the behavior is undefined. If any value put into the collection is not one understood by one of the callback functions, the behavior when that callback function is used is undefined.

If the dictionary will contain `CType` objects only, then pass a pointer to `kCTypeDictionaryValueCallBacks` as this parameter to use the default callback functions.

**Return Value**

A new dictionary, or `NULL` if there was a problem creating the object. Ownership follows the Create Rule in *Memory Management Programming Guide for Core Foundation*.

**Availability**

Available in iOS 2.0 and later.

**Declared In**

`CFDictionary.h`

## CFDictionaryCreateMutableCopy

Creates a new mutable dictionary with the key-value pairs from another dictionary.

```

CFMutableDictionaryRef CFDictionaryCreateMutableCopy (
    CFAllocatorRef allocator,
    CFIndex capacity,
    CFDictionaryRef theDict
);

```

### Parameters

*allocator*

The allocator to use to allocate memory for the new dictionary and its storage for key-value pairs. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

*capacity*

The maximum number of key-value pairs that can be contained by the new dictionary. The dictionary starts with the same number of key-value pairs as *theDict* and can grow to this number of values (and it can have less).

Pass `0` to specify that the maximum capacity is not limited. If non-`0`, *capacity* must be greater than or equal to the count of *theDict*.

*theDict*

The dictionary to copy. The keys and values from the dictionary are copied as pointers into the new dictionary, not that which the values point to (if anything). The keys and values are also retained by the new dictionary. The count of the new dictionary is the same as the count of *theDict*. The new dictionary uses the same callbacks as *theDict*.

### Return Value

A new dictionary that contains the same values as *theDict*. Ownership follows the Create Rule in *Memory Management Programming Guide for Core Foundation*.

### Availability

Available in iOS 2.0 and later.

### Declared In

CFDictionary.h

## CFDictionaryRemoveAllValues

Removes all the key-value pairs from a dictionary, making it empty.

```

void CFDictionaryRemoveAllValues (
    CFMutableDictionaryRef theDict
);

```

### Parameters

*theDict*

The dictionary to modify.

### Availability

Available in iOS 2.0 and later.

### Declared In

CFDictionary.h



## CFDictionaryRemoveValue

Removes a key-value pair.

```
void CFDictionaryRemoveValue (
    CFMutableDictionaryRef theDict,
    const void *key
);
```

### Parameters

*theDict*

The dictionary to modify.

*key*

The key of the value to remove from *theDict*. If a key which matches *key* is present in *theDict*, the key-value pair is removed from the dictionary, otherwise this function does nothing ("remove if present").

### Availability

Available in iOS 2.0 and later.

### Declared In

CFDictionary.h

## CFDictionaryReplaceValue

Replaces a value corresponding to a given key.

```
void CFDictionaryReplaceValue (
    CFMutableDictionaryRef theDict,
    const void *key,
    const void *value
);
```

### Parameters

*theDict*

The dictionary to modify.

*key*

The key of the value to replace in *theDict*. If a key which matches *key* is present in the dictionary, the value is changed to the *value*, otherwise this function does nothing ("replace if present").

*value*

The new value for *key*. The *value* object is retained by *theDict* using the retain callback provided when *theDict* was created, and the old value is released. *value* must be of the type expected by the retain and release callbacks.

### Availability

Available in iOS 2.0 and later.

### Declared In

CFDictionary.h

## CFDictionarySetValue

Sets the value corresponding to a given key.

```
void CFDictionarySetValue (
    CFMutableDictionaryRef theDict,
    const void *key,
    const void *value
);
```

**Parameters***theDict*

The dictionary to modify. If this parameter is a fixed-capacity dictionary and it is full before this operation, and the key does not exist in the dictionary, the behavior is undefined.

*key*

The key of the value to set in *theDict*. If a key which matches *key* is already present in the dictionary, only the value for the key is changed ("add if absent, replace if present"). If no key matches *key*, the key-value pair is added to the dictionary.

If a key-value pair is added, both *key* and *value* are retained by the dictionary, using the retain callback provided when *theDict* was created. *key* must be of the type expected by the key retain callback.

*value*

The value to add to or replace in *theDict*. *value* is retained using the value retain callback provided when *theDict* was created, and the previous value if any is released. *value* must be of the type expected by the retain and release callbacks.

**Availability**

Available in iOS 2.0 and later.

**Declared In**

CFDictionary.h

## Data Types

**CFMutableDictionaryRef**

A reference to a mutable dictionary object.

```
typedef struct __CFDictionary *CFMutableDictionaryRef;
```

**Availability**

Available in iOS 2.0 and later.

**Declared In**

CFDictionary.h

# Document Revision History

---

This table describes the changes to *CFMutableDictionary Reference*.

Date	Notes
2009-11-17	Clarified capacity argument in <code>CFDictionaryCreateMutable</code> and <code>CFDictionaryCreateMutableCopy</code> .
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-04-29	Moved Introduction to new Introduction page.
2004-10-05	Clarification of use of predefined callback structures in <a href="#">CFDictionaryCreateMutable</a> (page 6).
2003-08-01	Enhanced description of all the <code>kCFTType*Callbacks</code> and added link to Carbon-Cocoa integration document.
2003-01-01	First version of this document.

## REVISION HISTORY

### Document Revision History