

---

# ABMutableMultiValue Reference for C

[Apple Applications](#) > [Address Book](#)



2003-08-20



Apple Inc.  
© 2002, 2003 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, Mac, Mac OS, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

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

---

## **ABMutableMultiValue Reference for C 5**

---

Overview	5
Functions	5
ABMultiValueAdd	5
ABMultiValueCreateMutable	6
ABMultiValueInsert	7
ABMultiValueRemove	7
ABMultiValueReplaceLabel	8
ABMultiValueReplaceValue	8
ABMultiValueSetPrimaryIdentifier	9
Data Types	10
ABMutableMultiValueRef	10

---

## **Document Revision History 11**

---

## **Index 13**

---



# ABMutableMultiValue Reference for C

---

<b>Derived From:</b>	ABMultiValue : CType
<b>Framework:</b>	AddressBook/ABAddressBookC.h
<b>Companion guide</b>	Address Book Programming Guide for Mac OS X
<b>Declared in</b>	ABAddressBookC.h

## Overview

The `ABMultiValue` and `ABMutableMultiValue` opaque types are used to represent properties that might have multiple values. Each value in a multi-value list must be the same type, and has an associated pre-defined or user-defined label, and unique identifier. The labels, however, need not be unique. For example, you can have multiple “Home” phone numbers. Each multi-value object may have a primary identifier—used as a default value when a label is not provided. For example, a person record may have multiple addresses with the labels “Home” and “Work”, where “Work” is designated as the primary value. Instances of `ABMutableMultiValue` are mutable, see `ABMultiValue` for additional functions that access the content of a multi-value list.

You can use either the [ABMultiValueAdd](#) (page 5) or [ABMultiValueInsert](#) (page 7) functions to add value/label pairs to a multi-value list. You can remove an entry in a multi-value list using the [ABMultiValueRemove](#) (page 7) function. You can also replace values and labels using the [ABMultiValueReplaceLabel](#) (page 8) and [ABMultiValueReplaceValue](#) (page 8) functions.

Use the [ABMultiValueSetPrimaryIdentifier](#) (page 9) function to set the primary identifier—that is, designate the corresponding value as the default value for a multi-value list. Use the [ABMultiValueCopyIdentifierAtIndex](#) function to get the unique identifier for a value/label pair.

The `ABMutableMultiValue` opaque type is “toll-free bridged” with its Objective-C counterpart. This means that the `ABMutableMultiValueRef` type is interchangeable in function or method calls with instances of the `ABMutableMultiValue` class.

## Functions

### **ABMultiValueAdd**

Adds a value and its label to a multi-value list.

```
bool ABMultiValueAdd (
    ABMutableMultiValueRef multiValue,
    CTypeRef value,
    CFStringRef label,
    CFStringRef *outIdentifier
);
```

**Parameters***multiValue*

The multi-value list you wish to modify.

*value*

An object representing a value in a multi-value list—it must be of the correct type. For example, if *multiValue* is the value for a property of type `kABMultiStringProperty`, then *value* needs to be a `CFString` object. See [Property Types](#) for a list of supported types in a multi-value list (see descriptions of the `kABMulti...` constants). If *value* is `NULL`, this function raises an exception.

*label*

The label for *value*—it need not be unique. If *label* is `NULL`, this function raises an exception.

*outIdentifier*

If *value* is added successfully, this parameter returns the new identifier.

**Return Value**

`true` if successfully, `false` otherwise.

**Discussion**

This function performs no type checking and will let you add a value whose type does not match the types of the other values in the list. However, if you try to use a multi-value list whose values are not all of the same type, functions, such as the `ABRecord ABRecordSetValue` function, will return `NULL` or `kABErrorInProperty`.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

AddressBookCarbon

**Declared In**

ABAddressBookC.h

**ABMultiValueCreateMutable**

Returns a newly created mutable multi-value list object.

```
ABMutableMultiValueRef ABMultiValueCreateMutable (
    void
);
```

**Return Value**

A newly created `ABMutableMultiValue` object. You are responsible for releasing this object.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

AddressBookCarbon

**Declared In**

ABAddressBookC.h

**ABMultiValueInsert**

Inserts a value and its label at the given index in a multi-value list.

```
bool ABMultiValueInsert (
    ABMutableMultiValueRef multiValue,
    CTypeRef value,
    CFStringRef label,
    CFIndex index,
    CFStringRef *outIdentifier
);
```

**Parameters***multiValue*

The multi-value list you wish to modify.

*value*

An object representing a value in a multi-value list—it must be of the correct type. For example, if *multiValue* is the value for a property of type `kABMultiStringProperty`, then *value* needs to be a `CFString` object. See [Property Types](#) for a list of supported types in a multi-value list (see descriptions of the `kABMulti...` constants). If *value* is `NULL`, this function raises an exception.

*label*

The label for *value*—it need not be unique. If *label* is `NULL`, this function raises an exception.

*index*

The index to insert *value* at. If *index* is out of bounds, this function raises an exception.

*outIdentifier*

If *value* is added successfully, this parameter returns the new identifier.

**Return Value**

`true` if successfully, `false` otherwise.

**Discussion**

This function performs no type checking and will let you add a value whose type does not match the types of the other values in the list. However, if you try to use a multi-value list whose values are not all of the same type, functions, such as the `ABRecord ABRecordSetValue` function, will return `NULL` or `kABErrorProperty`.

**Version Notes****Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

ABAddressBookC.h

**ABMultiValueRemove**

Removes the value and label at the given index.

```
bool ABMultiValueRemove (
    ABMutableMultiValueRef multiValue,
    CFIndex index
);
```

**Parameters***multiValue*

The multi-value list you wish to modify.

*index*The index of the entry to be removed. If *index* is out of bounds, this function raises an exception.**Return Value**

true if successfully, false otherwise.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

ABAddressBookC.h

**ABMultiValueReplaceLabel**

Replaces the label at the given index.

```
bool ABMultiValueReplaceLabel (
    ABMutableMultiValueRef multiValue,
    CFStringRef label,
    CFIndex index
);
```

**Parameters***multiValue*

The multi-value list you wish to modify.

*label*The new label at *index*—it need not be unique. If *label* is NULL, this function raises an exception.*index*The index of the entry to be modified. If *index* is out of bounds, this function raises an exception.**Return Value**

true if successfully, false otherwise.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

ABAddressBookC.h

**ABMultiValueReplaceValue**

Replaces the value at the given index.



```
bool ABMultiValueReplaceValue (
    ABMutableMultiValueRef multiValue,
    CFTypeRef value,
    CFIndex index
);
```

**Parameters***multiValue*

The multi-value list you wish to modify.

*value*

An object representing the new value in a multi-value list—it must be of the correct type. For example, if *multiValue* is the value for a property of type `kABMultiStringProperty`, then *value* needs to be a `CFString` object. See [Property Types](#) for a list of supported types in a multi-value list (see descriptions of the `kABMulti...` constants). If *value* is `NULL`, this function raises an exception.

*index*

The index of the entry to be modified. If *index* is out of bounds, this function raises an exception.

**Return Value**

`true` if successfully, `false` otherwise.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

`ABAddressBookC.h`

**ABMultiValueSetPrimaryIdentifier**

Sets the primary value to be the value for the given identifier.

```
bool ABMultiValueSetPrimaryIdentifier (
    ABMutableMultiValueRef multiValue,
    CFStringRef identifier
);
```

**Parameters***multiValue*

The multi-value list you wish to modify.

*identifier*

The identifier corresponding to the value you wish to designate as the primary value for this multi-value list. Use the `ABMultiValueCopyIdentifierAtIndex` function to get the identifier given the index. If *identifier* is `NULL`, this function raises an exception.

**Return Value**

`true` if successfully, `false` otherwise.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

`ABAddressBookC.h`

## Data Types

### **ABMutableMultiValueRef**

A reference to an ABMutableMultiValue object.

```
typedef struct __ABMultiValue *ABMutableMultiValueRef;
```

### **Availability**

Available in Mac OS X v10.2 and later.

### **Declared In**

ABAddressBookC.h

# Document Revision History

---

This table describes the changes to *ABMutableMultiValue Reference for C*.

Date	Notes
2003-08-20	Revised for Mac OS X v10.3.
2003-03-01	First version of the <i>ABMutableMultiValue Reference for C</i> .

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

- ABMultiValueAdd **function** [5](#)
- ABMultiValueCreateMutable **function** [6](#)
- ABMultiValueInsert **function** [7](#)
- ABMultiValueRemove **function** [7](#)
- ABMultiValueReplaceLabel **function** [8](#)
- ABMultiValueReplaceValue **function** [8](#)
- ABMultiValueSetPrimaryIdentifier **function** [9](#)
- ABMutableMultiValueRef **data type** [10](#)