
ABMutableMultiValue Class Reference

Data Management: Contact Data



2009-07-22



Apple Inc.
© 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, Cocoa, 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 Class Reference 5

Overview 5

Tasks 5

 Adding a Value 5

 Replacing Values and Labels 6

 Removing Values 6

 Setting the Primary Identifier 6

Instance Methods 6

 addValue:withLabel: 6

 insertValue:withLabel:atIndex: 7

 removeValueAndLabelAtIndex: 8

 replaceLabelAtIndex:withLabel: 8

 replaceValueAtIndex:withValue: 8

 setPrimaryIdentifier: 9

Document Revision History 11

Index 13

ABMutableMultiValue Class Reference

Inherits from	ABMultiValue : NSObject
Conforms to	NSCopying (ABMultiValue) NSMutableCopying (ABMultiValue) NSFastEnumeration (ABMultiValue) NSObject (NSObject)
Framework	/System/Library/Frameworks/AddressBook.framework
Availability	Available in Mac OS X v10.2 and later.
Companion guide	Address Book Programming Guide for Mac OS X
Declared in	ABMultiValue.h
Related sample code	AddressBookCocoa

Overview

The `ABMultiValue` and `ABMutableMultiValue` classes are used to represent properties that might have multiple values. Each value in a multivalue list must be of the same type, and must have an associated predefined or user-defined label, and unique identifier. The labels, however, need not be unique. For example, you can have multiple Home phone numbers. Each multivalue 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 methods that access the content of a multivalue list.

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

Tasks

Adding a Value

- `addValue:withLabel:` (page 6)
Adds a value and its label to a multivalue list.

- [insertValue:withLabel:atIndex:](#) (page 7)
Inserts a value and its label at the given index in a multivalue list.

Replacing Values and Labels

- [replaceLabelAtIndex:withLabel:](#) (page 8)
Replaces the label at the given index.
- [replaceValueAtIndex:withValue:](#) (page 8)
Replaces the value at the given index.

Removing Values

- [removeValueAndLabelAtIndex:](#) (page 8)
Removes the value and label at the given index.

Setting the Primary Identifier

- [setPrimaryIdentifier:](#) (page 9)
Sets the primary value to be the value for the given identifier.

Instance Methods

addValue:withLabel:

Adds a value and its label to a multivalue list.

```
(NSString *)addValue:(id)value withLabel:(NSString *)label
```

Parameters

value

The value to add.

label

The label to associate with the value.

Return Value

The new identifier if *value* is added successfully; otherwise, `nil`.

Discussion

The *value* argument must be of the correct type. For example, if the receiver is the value for a property of type `kABMultiStringProperty`, then *value* needs to be an `NSString` object. See “Property Types” in *Address Book Objective-C Constants Reference* for a list of supported types in a multivalue list (see descriptions of the `kABMulti...` constants). If either the *value* or the *label* argument is `nil`, this method raises an exception.

This method 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 multivalue list whose values are not all of the same type, other methods, such as the `ABRecord setValue:forProperty:` method, will return `NO` or `kABErrorInProperty`.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [insertValue:withLabel:atIndex:](#) (page 7)

Related Sample Code

AddressBookCocoa

Declared In

ABMultiValue.h

insertValue:withLabel:atIndex:

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

```
- (NSString *)insertValue:(id)value withLabel:(NSString *)label
  atIndex:(NSUInteger)index
```

Parameters

value

The value to add.

label

The label to associate with the value.

index

The index where the value will be inserted.

Return Value

The identifier of the inserted value and label if they are added successfully; otherwise, `nil`.

Discussion

If either the value or the label is `nil` or if the index is out of bounds, this method raises an exception

This method 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 multivalue list whose values are not all of the same type, other methods, such as the `ABRecord setValue:forProperty:` method, will return `NO` or `kABErrorInProperty`.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [addValue:withLabel:](#) (page 6)

Declared In

ABMultiValue.h

removeValueAndLabelAtIndex:

Removes the value and label at the given index.

- (BOOL)removeValueAndLabelAtIndex:(NSUInteger) *index*

Parameters

index

The index of the value-label pair that will be removed.

Return Value

YES if successful; otherwise NO.

Discussion

If the index is out of bounds, this method raises an exception.

Availability

Available in Mac OS X v10.2 and later.

Declared In

ABMultiValue.h

replaceLabelAtIndex:withLabel:

Replaces the label at the given index.

- (BOOL)replaceLabelAtIndex:(NSUInteger) *index* withLabel:(NSString *) *label*

Parameters

index

The index of the label that will be replaced.

label

The new label.

Return Value

YES if successful; otherwise, NO.

Discussion

If the label is `nil` or if the index is out of bounds, this method raises an exception.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [replaceValueAtIndex:withValue:](#) (page 8)

Declared In

ABMultiValue.h

replaceValueAtIndex:withValue:

Replaces the value at the given index.

- (BOOL)replaceValueAtIndex:(NSUInteger) *index* withValue:(id) *value*

Parameters

index

The index of the value that will be replaced.

value

The new value.

Return Value

YES if successful; otherwise NO.

Discussion

If the value is `nil` or if the index is out of bounds, this method raises an exception.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [replaceLabelAtIndex:withLabel:](#) (page 8)

Declared In

ABMultiValue.h

setPrimaryIdentifier:

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

```
- (BOOL)setPrimaryIdentifier:(NSString *)identifier
```

Parameters

identifier

The identifier whose value will be used as the primary value for a multivalue property.

Return Value

YES if successful; otherwise NO.

Discussion

If the identifier is `nil`, this method raises an exception. Use the `identifierAtIndex:` method to get the identifier given the index.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `primaryIdentifier` (ABMultiValue)

Declared In

ABMultiValue.h

Document Revision History

This table describes the changes to *ABMutableMultiValue Class Reference*.

Date	Notes
2009-07-22	Added descriptions of parameters and return values to API symbols, as needed.
2009-06-02	Added descriptions of parameters and return values to API symbols, as needed.
2006-05-23	First publication of this content as a separate document.
	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`addValue:withLabel:` **instance method** [6](#)

I

`insertValue:withLabel:atIndex:` **instance method** [7](#)

R

`removeValueAndLabelAtIndex:` **instance method** [8](#)
`replaceLabelAtIndex:withLabel:` **instance method** [8](#)
`replaceValueAtIndex:withValue:` **instance method** [8](#)

S

`setPrimaryIdentifier:` **instance method** [9](#)