Address Book Objective-C Framework Reference

Cocoa > Apple Applications

2007-07-08

Contents

Introduction

Framework	/System/Library/Frameworks/AddressBook.framework
Header file directories	/System/Library/Frameworks/AddressBook.framework/Headers
Declared in	ABActions.h ABAddressBook.h ABGlobals.h ABGroup.h ABImageLoading.h ABMultiValue.h ABPeoplePickerView.h ABPerson.h ABRecord.h ABSearchElement.h ABTypedefs.h

The Address Book is a centralized database for contact and other personal information for people. Users need to enter personal information about themselves and their friends only once, instead of entering it repeatedly whenever the information is used. Applications that support the Address Book framework share this contact information with other applications, include Apple's Mail and iChat.

Classes

ABAddressBook Class Objective-C Reference

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

Overview

The ABAddressBook class provides a programming interface to the Address Book—a centralized database used by multiple applications to store contact and other personal information about people. The Address Book database also supports the notion of a "group" containing one or more persons. People may belong to multiple groups, and groups may also belong to other groups with some restrictions (for example, no circular references are allowed).

The ABAddressBook class provides methods for accessing, adding, and removing group and person records including the "me" record corresponding to the logged-in user. For example, you use the `groups` (page 15) method to get an array of all the group records in the database, or the `people` (page 17) method to get all the person records. You use the `me` (page 16) method to get the person record corresponding to the logged-in user. You can also add and remove records using the `addRecord:` (page 14) and `removeRecord:` (page 18) methods.

You can also search for records matching a particular query you specify by creating an ABSearchElement object. You use the `searchElementForProperty:label:key:value:comparison:` (page 63) method in the ABPerson and ABGroup class to create an ABSearchElement object for the corresponding record. Then use the `recordsMatchingSearchElement:` (page 18) ABAddressBook method, passing the ABSearchElement as the argument, to query the database. See `ABSearchElement` for more methods that create compound queries.

Your application uses a shared instance of ABAddressBook returned by the `sharedAddressBook` (page 14) class method to interact with the database (multiple ABAddressBook instances are not supported). Changes you make to record objects are stored in memory and saved to disk when you invoke the `saveAndReturnError:` (page 19) method.

The Address Book posts notifications if any application including yours makes changes to the database. Typically, you observe these notifications to update any dependent view or model objects in your application. Use NSNotificationCenter to register for the ABAddressBook notifications: `kABDatabaseChangedNotification` (page 20) and `kABDatabaseChangedExternallyNotification` (page 20). These notifications are not sent until the`sharedAddressBook` class method has been invoked.

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

Tasks

Creating and Initializing an ABAddressBook

`+ sharedAddressBook` (page 14)

`+ addressBook` (page 13)
  Returns a new instance of ABAddressBook.

Retrieving Groups and People

`- groups` (page 15)

`- people` (page 17)

Setting and Retrieving the Logged-in User’s Record

`- me` (page 16)

`- setMe:` (page 19)

Retrieving a Specific Record

`- recordForUniqueId:` (page 17)

Retrieving the Class of a Record

`- recordClassFromUniqueId:` (page 17)

Retrieving a Formatted Address

`- formattedAddressFromDictionary:` (page 15)

Retrieving Default Values

`- defaultCountryCode` (page 15)

`- defaultNameOrdering` (page 15)

Adding and Removing Records

`- addRecord:` (page 14)

`- removeRecord:` (page 18)

Searching

`- recordsMatchingSearchElement:` (page 18)

Saving and Detecting Changes

`- hasUnsavedChanges` (page 16)

`- save` (page 18)

`- saveAndReturnError:` (page 19)
Saves all the changes made since the last save.

Class Methods

addressBook

Returns a new instance of ABAddressBook.

```
+ (ABAddressBook *)addressBook
```

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ sharedAddressBook`

Declared In
`ABAddressBook.h`

sharedAddressBook

```
+ (ABAddressBook *)sharedAddressBook
```

Discussion
Returns the unique shared instance of ABAddressBook. This method returns the address book for the logged-in user that is shared by every application. If you call this method more than once or try to create a new address book, you get a pointer to the same shared address book.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ addressBook`

Related Sample Code
ABPresence
AddressBookCocoa
Birthdays

Declared In
`ABAddressBook.h`

Instance Methods

addRecord:

```
- (BOOL)addRecord:(ABRecord *)record
```

Discussion
Adds an ABPerson or ABGroup record to the Address Book database. If the *`record`* argument is `nil`, this method raises an exception. This method returns `YES` if the record was added successfully, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- removeRecord:` (page 18)
`- initWithVCardRepresentation:` (page 65) (ABPerson)

Declared In
`ABAddressBook.h`

defaultCountryCode

```
- (NSString *)defaultCountryCode
```

Discussion
Returns the default country code for records with unspecified country codes.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABAddressBook.h`

defaultNameOrdering

```
- (NSInteger)defaultNameOrdering
```

Discussion
Returns the default name ordering defined by the user in the Address Book preferences. The possible values are `kABFirstNameFirst` and `kABLastNameFirst`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABAddressBook.h`

formattedAddressFromDictionary:

```
- (NSAttributedString *)formattedAddressFromDictionary:(NSDictionary *)address
```

Discussion
Returns an attributed string containing the formatted address. The string's attributes match address dictionary keys, such as `kABAddressStreetKey`. Each attribute value contains the localized description of the key. (For example, the value of a Canadian `kABAddressZIPKey` field would be "Postal Code", while the value of a French one would be "Code Postale".)

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABAddressBook.h`

groups

```
- (NSArray *)groups
```

Discussion

Returns an array of all the groups in the Address Book database, or returns an empty array if the database doesn't contain any groups

Availability

Available in Mac OS X v10.2 and later.

See Also

`- people` (page 17)

Declared In

`ABAddressBook.h`

hasUnsavedChanges

```
- (BOOL)hasUnsavedChanges
```

Discussion

Returns `YES` if there are unsaved changes, `NO` otherwise. The unsaved changes flag is set automatically whenever changes are made.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- saveAndReturnError:` (page 19)

Declared In

`ABAddressBook.h`

me

```
- (ABPerson *)me
```

Discussion

Returns the ABPerson record that represents the logged-in user, or `nil` if the user never specified such a record.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- setMe:` (page 19)

Related Sample Code

Birthdays

Declared In

`ABAddressBook.h`

people

```
- (NSArray *)people
```

Discussion
Returns an array of all the people in the Address Book database, or an empty array if the database doesn't contain any people.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- groups` (page 15)

Related Sample Code
ABPresence

Declared In
`ABAddressBook.h`

recordClassFromUniqueId:

```
- (NSString *)recordClassFromUniqueId:(NSString *)uniqueId
```

Discussion
Returns the class name of the record that matches the given unique ID.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABAddressBook.h`

recordForUniqueId:

```
- (ABRecord *)recordForUniqueId:(NSString *)uniqueId
```

Discussion
Returns the ABPerson or ABGroup record that matches the given unique ID, or `nil` if no record has the given ID. If *`uniqueId`* is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- uniqueId` (page 78) (ABRecord)

Declared In
`ABAddressBook.h`

recordsMatchingSearchElement:

```
- (NSArray *)recordsMatchingSearchElement:(ABSearchElement *)search
```

Discussion
Returns an array of records that match the given search element, or returns an empty array if no records match the search element. If `search` is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ searchElementForProperty:label:key:value:comparison:` (page 63) (ABPerson)
`+ searchElementForProperty:label:key:value:comparison:` (page 24) (ABGroup)
`+ searchElementForConjunction:children:` (page 82) (ABSearchElement)

Related Sample Code
AddressBookCocoa
Birthdays

Declared In
`ABAddressBook.h`

removeRecord:

```
- (BOOL)removeRecord:(ABRecord *)record
```

Discussion
Removes an ABPerson or ABGroup record from the Address Book database. If `record` is `nil`, this method raises an exception. This method returns `YES` if the record was removed successfully, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- addRecord:` (page 14)

Declared In
`ABAddressBook.h`

save

```
- (BOOL)save
```

Discussion
Saves all the changes made since the last save. Returns `YES` if this method is successful or there were no changes, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also

- `hasUnsavedChanges` (page 16)
- `saveAndReturnError:` (page 19)

Declared In

`ABAddressBook.h`

saveAndReturnError:

Saves all the changes made since the last save.

```
- (BOOL)saveAndReturnError:(NSError **)error
```

Parameters

error

A pointer to an error object that is set to an NSError instance if an error occurs.

Return Value

`YES` if successful or there were no changes; otherwise, `NO`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `save` (page 18)
- `hasUnsavedChanges` (page 16)

Declared In

`ABAddressBook.h`

setMe:

```
- (void)setMe:(ABPerson *)person
```

Discussion

Sets the record that represents the logged-in user. If you don't want a record to represent the logged-in user, then pass `nil` as the *person* argument.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `me` (page 16)

Declared In

`ABAddressBook.h`

Notifications

These notifications are sent when something changes in the Address Book database. These are not sent until the`sharedAddressBook` class method has been invoked.

kABDatabaseChangedNotification

Posted when this process has changed the Address Book database. The process id of the sender and the effective user ID are always included in the user-info dictionary; you access them through the keys `"kABSenderProcessID"` and `"kABUserUID"`, respectively. In addition, depending on the operation performed on the address book, one or more of the following keys may be included: `kABInsertedRecords`, `kABUpdatedRecords`, and `kABDeletedRecords`. The values for each of the keys are the unique IDs of the records that were inserted, updated, or deleted, respectively. If the values for all the keys are `nil`, everything has changed, such as when the Address Book database is restored from a backup copy.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABGlobals.h`

kABDatabaseChangedExternallyNotification

Posted when a process other than the current one has changed the Address Book database. The process id of the sender and the effective user ID are always included in the user-info dictionary; you access them through the keys `"kABSenderProcessID"` and `"kABUserUID"`, respectively. In addition, depending on the operation performed on the address book, one or more of the following keys may be included: `kABInsertedRecords`, `kABUpdatedRecords`, and `kABDeletedRecords`. The values for each of the keys are the unique IDs of the records that were inserted, updated, or deleted, respectively. If the values for all the keys are `nil`, everything has changed, such as when the Address Book database is restored from a backup copy.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABGlobals.h`

ABGroup Class Objective-C Reference

Inherits from	ABRecord : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AddressBook.framework
Availability	Available in Mac OS X v10.2 and later.
Declared in	ABGroup.h
Companion guide	Address Book Programming Guide for Mac OS X

Overview

The ABGroup class supports the concept of a "group" containing one or more persons. People may belong to multiple groups, and groups may also belong to other groups as long as the relationship does not cause a circular reference. The only predefined property of a group is its name. However, similar to person records, you can add your own properties to group records. Groups not only help to organize person records, but allow you to create email distribution lists.

Use the `members` (page 26) method to get all the members of a group, use the `addMember:` (page 25) method to add people to a group, and the `removeMember:` (page 27) method to remove people from a group. Use the `addSubgroup:` (page 25) method to create a subgroup.

Use the `addPropertiesAndTypes:` (page 23) method to add additional program-defined properties to group records. Because the Address Book database is stored as a property list, these program-defined properties may be ignored by other applications. Note that the Address Book database is accessed by multiple application and is not encrypted so your application should not store any sensitive information in the database.

You can also search for records matching a particular "query" you specify by creating an ABSearchElement object. Use the `searchElementForProperty:label:key:value:comparison:` (page 24) method to create an ABSearchElement object containing your query. Then use the ABAddressBook `recordsMatchingSearchElement:` (page 18) method, passing the ABSearchElement as the argument, to query the database. See ABSearchElement for methods that create compound queries.

The ABGroup class is "toll-free bridged" with its procedural C, opaque type, counterpart. This means that the `ABGroupRef` type is interchangeable in function or method calls with instances of the ABGroup class.

Tasks

Managing Properties

`+ addPropertiesAndTypes:` (page 23)

`+ removeProperties:` (page 23)

`+ properties` (page 23)

`+ typeOfProperty:` (page 24)

Managing Persons

`- addMember:` (page 25)

`- removeMember:` (page 27)

`- members` (page 26)

Managing Subgroups

`- addSubgroup:` (page 25)

`- removeSubgroup:` (page 27)

`- parentGroups` (page 26)

`- subgroups` (page 28)

Distribution Lists

`- distributionIdentifierForProperty:person:` (page 26)

`- setDistributionIdentifier:forProperty:person:` (page 28)

Searching

`+ searchElementForProperty:label:key:value:comparison:` (page 24)

Class Methods

addPropertiesAndTypes:

```
+ (NSInteger)addPropertiesAndTypes:(NSDictionary *)properties
```

Discussion
Adds the given properties to all records of this type in the Address Book database, and returns the number of properties successfully added. In each dictionary entry, the key is a string with the property's name, and the value is a constant with the property's type. The property's name must be unique. You may want to use Java-style package names for your properties, for example, `"org.dogclub.dogname"` or `"com.mycompany.groupID"`. The property type must be one of the constants described in "Constants" (page 78) in ABRecord.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ removeProperties:` (page 23)

Declared In
`ABGroup.h`

properties

```
+ (NSArray *)properties
```

Discussion
Returns an array of the names of all the properties for this record in the Address Book database.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ typeOfProperty:` (page 24)

Declared In
`ABGroup.h`

removeProperties:

```
+ (NSInteger)removeProperties:(NSArray *)properties
```

Discussion

Removes the given properties from all the records of this type in the Address Book database, and returns the number of properties successfully removed.

Availability

Available in Mac OS X v10.2 and later.

See Also

`+ addPropertiesAndTypes:` (page 23)

Declared In

`ABGroup.h`

searchElementForProperty:label:key:value:comparison:

```
+ (ABSearchElement *)searchElementForProperty:(NSString *)property label:(NSString
      *)label key:(NSString *)key value:(id)value
     comparison:(ABSearchComparison)comparison
```

Discussion

Returns a search element object that searches for records of this type.

- *`property`* is the name of the property to search on. It cannot be `nil`. For a full list of the properties, see "Constants" (page 28) and "Constants" (page 78) in ABRecord.
- *`label`* is the label name for a multi-value list. If *`property`* does not have multiple values, pass `nil`. If *`property`* does have multiple values, pass `nil` to search all the values. By default, ABGroup records don't contain any multi-value list properties.
- *`key`* is the key name for a dictionary. If *`property`* is not a dictionary, pass `nil`. If *`property`* is a dictionary, pass `nil` to search all keys. By default, ABGroup records don't contain any properties that are dictionaries.
- *`value`* is what you're searching for. If `nil`, then the only supported value for *`comparison`* is `kABEqual` or `kABNotEqual`.
- *`comparison`* specifies the type of comparison to perform and is an `ABSearchComparison` (page 102), such as `kABEqual` or `kABPrefixMatchCaseInsensitive`.

Availability

Available in Mac OS X v10.2 and later.

See Also

`+ searchElementForProperty:label:key:value:comparison:` (page 63) (ABPerson)
`+ searchElementForConjunction:children:` (page 82) (ABSearchElement)
`- recordsMatchingSearchElement:` (page 18) (ABAddressBook)

Declared In

`ABGroup.h`

typeOfProperty:

```
+ (ABPropertyType)typeOfProperty:(NSString *)property
```

Discussion
Returns the type for a given property. If the property does not exist, this method returns `kABErrorInProperty`.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ properties` (page 23)

Declared In
`ABGroup.h`

Instance Methods

addMember:

```
- (BOOL)addMember:(ABPerson *)person
```

Discussion
Adds a person to a group. Returns `YES` if successful. If the *`person`* argument is already part of the group, this method does nothing and returns `NO`. If *`person`* is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- removeMember:` (page 27)
`- addSubgroup:` (page 25)
`- members` (page 26)

Declared In
`ABGroup.h`

addSubgroup:

```
- (BOOL)addSubgroup:(ABGroup *)group
```

Discussion
Adds a subgroup to another group. Returns `YES` if successful. If the *`group`* argument is already part of the receiver, this method does nothing and returns `NO`. If adding the group would create a recursion, this method also does nothing and returns `NO`. (For example, if the group "Animal Lovers" is in "Dog Lovers," and you add "Dog Lovers" to "Animal Lovers," that would create a recursion, which this method won't allow.) If the *`group`* argument is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- removeSubgroup:` (page 27)

- `addMember:` (page 25)
- `subgroups` (page 28)

Declared In
`ABGroup.h`

distributionIdentifierForProperty:person:

```
- (NSString *)distributionIdentifierForProperty:(NSString *)property person:(ABPerson
      *)person
```

Discussion
Returns the distribution identifier for the given property and person. If a distribution identifier is not set, this method returns the multi-value's primary identifier. If either the *`property`* or *`person`* arguments are `nil`, this method returns `nil`. Also, returns `nil` if *`property`* is not a multi-value list property.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `setDistributionIdentifier:forProperty:person:` (page 28)
- `primaryIdentifier` (page 34) (ABMultiValue)

Declared In
`ABGroup.h`

members

```
- (NSArray *)members
```

Discussion
Returns an array of persons in a group. If this group doesn't contain any people, this method returns an empty array.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `addMember:` (page 25)
- `removeMember:` (page 27)
- `subgroups` (page 28)

Declared In
`ABGroup.h`

parentGroups

```
- (NSArray *)parentGroups
```

Discussion
Returns an array containing a group's parents—the groups that a group belongs to. If this group doesn't belong to any groups, this method returns an empty array.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `subgroups` (page 28)

Declared In
`ABGroup.h`

removeMember:

```
- (BOOL)removeMember:(ABPerson *)person
```

Discussion
Removes a person from a group. Returns `YES` if successful. If the *`person`* argument is not in the group, this method does nothing and returns `NO`. If *`person`* is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `addMember:` (page 25)
- `members` (page 26)
- `removeSubgroup:` (page 27)

Declared In
`ABGroup.h`

removeSubgroup:

```
- (BOOL)removeSubgroup:(ABGroup *)group
```

Discussion
Removes a subgroup from a group. Returns `YES` if successful. If the *`group`* argument is not a subgroup, this method does nothing and returns `NO`. If *`group`* is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `addSubgroup:` (page 25)
- `subgroups` (page 28)
- `removeMember:` (page 27)

Declared In
`ABGroup.h`

setDistributionIdentifier:forProperty:person:

```
- (BOOL)setDistributionIdentifier:(NSString *)identifier forProperty:(NSString
    *)property person:(ABPerson *)person
```

Discussion
Assigning a specific distribution identifier for a person's multi-value list property so that the group can be used as a distribution list (mailing list, in the case of an email property). The default distribution identifier is a multi-value list's primary identifier. Use this method if you need to change the distribution identifier for a particular person. For example, if the default identifier is a person's home email but you want to use John's work email, invoke this method passing `kABEmailWorkLabel` as the *`identifier`* argument, `kABEmailProperty` as the *`property`* argument, and John's person object as the *`person`* argument. Pass `nil` for the *`identifier`* argument to reset the distribution identifier to its default. If *`person`* is `nil`, this method raises an exception. Returns `YES` if successful, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

See Also
`– distributionIdentifierForProperty:person:` (page 26)
`– setPrimaryIdentifier:` (page 40) (ABMutableMultiValue)

Declared In
`ABGroup.h`

subgroups

```
- (NSArray *)subgroups
```

Discussion
Returns an array containing a group's subgroups. If this group doesn't contain any groups, this method returns an empty array.

Availability
Available in Mac OS X v10.2 and later.

See Also
`– addSubgroup:` (page 25)
`– removeSubgroup:` (page 27)
`– members` (page 26)

Declared In
`ABGroup.h`

Constants

This is one of the properties that an ABGroup record contains by default.ABRecord describes some properties that all records contain, in "Constants" (page 78)

Constant	Description
`kABGroupNameProperty`	Name of the group

ABMultiValue Class Objective-C Reference

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

Overview

The ABMultiValue and ABMutableMultiValue classes 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 this class are immutable, see ABMutableMultiValue for methods that manipulate the content of a multi-value list.

You can access values using a numeric index (similar to an array). Use the `identifierAtIndex:` (page 33) method to get an identifier, the `labelAtIndex:` (page 33) method to get a label, and the `valueAtIndex:` (page 35) method to get a value. However, a numeric index is temporary since a multi-value list may change. Each value or entry in a multi-value list has a unique identifier which can be used to save a reference to a specific value—the identifier is guaranteed never to change.

Use the `primaryIdentifier` (page 34) method to get the primary identifier (the identifier associated with the primary value).

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

Accessing the Primary Identifier

`– primaryIdentifier` (page 34)

Accessing Identifiers

`– identifierAtIndex:` (page 33)

`– indexForIdentifier:` (page 33)

Accessing Entries

`– labelAtIndex:` (page 33)

`– valueAtIndex:` (page 35)

`– valueForIdentifier:` (page 35)

`– labelForIdentifier:` (page 34)

Querying the List

`– count` (page 32)

`– propertyType` (page 34)

Instance Methods

count

```
- (NSUInteger)count
```

Discussion
Returns the number of entries in a multi-value list.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABMultiValue.h`

identifierAtIndex:

```
- (NSString *)identifierAtIndex:(NSUInteger)index
```

Discussion
Returns the identifier for the given index. If the *`index`* argument is out of bounds, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- indexForIdentifier:` (page 33)

Declared In
`ABMultiValue.h`

indexForIdentifier:

```
- (NSUInteger)indexForIdentifier:(NSString *)identifier
```

Discussion
Returns the index for the given identifier. Returns `NSNotFound` if the identifier is not found.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- identifierAtIndex:` (page 33)

Declared In
`ABMultiValue.h`

labelAtIndex:

```
- (NSString *)labelAtIndex:(NSUInteger)index
```

Discussion
Returns the label for the given index. If the *`index`* argument is out of bounds, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- labelForIdentifier:` (page 34)

Declared In
`ABMultiValue.h`

labelForIdentifier:

```
- (id)labelForIdentifier:(NSString *)identifier
```

Discussion
Returns the label for the given identifier. Returns `nil` if the identifier is not found.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– labelAtIndex:` (page 33)

Declared In
`ABMultiValue.h`

primaryIdentifier

```
- (NSString *)primaryIdentifier
```

Discussion
Returns the identifier for the primary value.

Availability
Available in Mac OS X v10.2 and later.

See Also
`– indexForIdentifier:` (page 33)
`– setPrimaryIdentifier:` (page 40) (ABMutableMultiValue)

Declared In
`ABMultiValue.h`

propertyType

```
- (ABPropertyType)propertyType
```

Discussion
Returns the type for the values in a multi-value list. If the multi-value list is empty or its values are of different types, it returns `kABErrorInProperty`.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABMultiValue.h`

valueAtIndex:

```
- (id)valueAtIndex:(NSUInteger)index
```

Discussion
Returns the value for the given index. If the `index` argument is out of bounds, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`– valueForIdentifier:` (page 35)

Declared In
`ABMultiValue.h`

valueForIdentifier:

```
- (id)valueForIdentifier:(NSString *)identifier
```

Discussion
Returns the value for the given identifier. Returns `nil` if the identifier is not found.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– valueAtIndex:` (page 35)

Declared In
`ABMultiValue.h`

ABMutableMultiValue Class Objective-C 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.
Declared in	ABMutliValue.h
Companion guide	Address Book Programming Guide for Mac OS X
Related sample code	AddressBookCocoa

Overview

The ABMultiValue and ABMutableMultiValue classes 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 methods that access the content of a multi-value list.

You can use either the `addValue:withLabel:` (page 38) or `insertValue:withLabel:atIndex:` (page 39) methods to add value/label pairs to a multi-value list. You can remove an entry in a multi-value list using the `removeValueAndLabelAtIndex:` (page 39) method. You can also replace values and labels using the `replaceLabelAtIndex:withLabel:` (page 40)and `replaceValueAtIndex:withValue:` (page 40) methods.

Use the `setPrimaryIdentifier:` (page 40) method to set the primary identifier—that is, designate the corresponding value as the default value for a multi-value list. Use the `identifierAtIndex:` (page 33) method to get the unique identifier for a value/label pair.

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

Tasks

Adding a Value

– `addValue:withLabel:` (page 38)

– `insertValue:withLabel:atIndex:` (page 39)

Replacing Values and Labels

– `replaceLabelAtIndex:withLabel:` (page 40)

– `replaceValueAtIndex:withValue:` (page 40)

Removing Values

– `removeValueAndLabelAtIndex:` (page 39)

Primary Identifier

– `setPrimaryIdentifier:` (page 40)

Instance Methods

addValue:withLabel:

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

Discussion
Adds a value and its label to a multi-value list. 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 "Constants" (page 78) for a list of supported types in a multi-value list (see descriptions of the `kABMulti...` constants). If *`value`* is added successfully, this method returns the new identifier, `nil` otherwise. If either the *`value`* or the *`label`* arguments are `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 multi-value list whose values are not all of the same type, other methods, such as the ABRecord `setValue:forProperty:` (page 77) method, will return `NO` or `kABErrorInProperty`.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- insertValue:withLabel:atIndex:` (page 39)

Related Sample Code
AddressBookCocoa

Declared In
`ABMultiValue.h`

insertValue:withLabel:atIndex:

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

Discussion
Inserts a value and its label at the given index in a multi-value list. If successful, this method returns the identifier, `nil` otherwise. 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 multi-value list whose values are not all of the same type, other methods, such as the ABRecord `setValue:forProperty:` (page 77) method, will return `NO` or `kABErrorInProperty`.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- addValue:withLabel:` (page 38)

Declared In
`ABMultiValue.h`

removeValueAndLabelAtIndex:

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

Discussion
Removes the value and label at the given index. Returns `YES` if successful, `NO` otherwise. 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:

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

Discussion
Replaces the label at the given index. Returns `YES` if successful, `NO` otherwise. 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 40)

Declared In
`ABMultiValue.h`

replaceValueAtIndex:withValue:

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

Discussion
Replaces the value at the given index. Returns `YES` if successful, `NO` otherwise. 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 40)

Declared In
`ABMultiValue.h`

setPrimaryIdentifier:

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

Discussion
Sets the primary value to be the value for the given identifier. Use the `identifierAtIndex:` (page 33) method to get the identifier given the index. Returns `YES` if successful, `NO` otherwise. If the identifier is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- primaryIdentifier` (page 34) (ABMultiValue)

Declared In
`ABMultiValue.h`

ABPeoplePickerView Class Objective-C Reference

Inherits from	NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AddressBook.framework
Declared in	ABPeoplePickerView.h
Availability	Available in Mac OS X v10.3 and later
Companion guide	Address Book Programming Guide for Mac OS X
Related sample code	CocoaPeoplePicker

Overview

The ABPeoplePickerView class allows you to customize the behavior of people-picker views in an application's user interface.

Tasks

Properties in Record List

– `addProperty:` (page 44)

– `columnTitleForProperty:` (page 45)

– `displayedProperty` (page 47)

– `properties` (page 48)

– `removeProperty:` (page 49)

– `setColumnTitle:forProperty:` (page 53)

- `setDisplayedProperty:` (page 53)

Multi-value Selection Behavior

- `setValueSelectionBehavior:` (page 54)
- `valueSelectionBehavior` (page 55)

Group and Record Selection

- `allowsGroupSelection` (page 44)
- `allowsMultipleSelection` (page 44)
- `deselectAll:` (page 46)
- `deselectGroup:` (page 46)
- `deselectIdentifier:forPerson:` (page 46)
- `deselectRecord:` (page 47)
- `selectedGroups` (page 49)
- `selectedIdentifiersForPerson:` (page 49)
- `selectedRecords` (page 50)
- `selectedValues` (page 50)
- `selectGroup:byExtendingSelection:` (page 50)
- `selectIdentifier:forPerson:byExtendingSelection:` (page 50)
- `selectRecord:byExtendingSelection:` (page 51)
- `setAllowsGroupSelection:` (page 52)
- `setAllowsMultipleSelection:` (page 52)

Accessory View

- `accessoryView` (page 43)
- `setAccessoryView:` (page 51)

Actions

- `clearSearchField:` (page 45)
- `editInAddressBook:` (page 47)
- `groupDoubleAction` (page 48)
- `nameDoubleAction` (page 48)
- `selectInAddressBook:` (page 51)
- `setGroupDoubleAction:` (page 53)
- `setNameDoubleAction:` (page 54)
- `setTarget:` (page 54)
- `target` (page 55)

User-settings Persistence

- `autosaveName` (page 45)
- `setAutosaveName:` (page 52)

Instance Methods

accessoryView

```
- (NSView *)accessoryView
```

Discussion
Returns the current accessory view.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAccessoryView:` (page 51)

Declared In
`ABPeoplePickerView.h`

addProperty:

```
- (void)addProperty:(NSString *)property
```

Discussion
Adds a property to the group of properties whose values are shown in the record list.

For additional information about properties see "Person Properties" (page 107)

Availability
Available in Mac OS X v10.3 and later.

See Also
`- removeProperty:` (page 49)
`- properties` (page 48)

Declared In
`ABPeoplePickerView.h`

allowsGroupSelection

```
- (BOOL)allowsGroupSelection
```

Discussion
Returns the current group-selection setting.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAllowsGroupSelection:` (page 52)

Declared In
`ABPeoplePickerView.h`

allowsMultipleSelection

```
- (BOOL)allowsMultipleSelection
```

Discussion
Returns the current multiple-selection selection setting.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAllowsMultipleSelection:` (page 52)

Declared In
`ABPeoplePickerView.h`

autosaveName

```
- (NSString *)autosaveName
```

Discussion
Returns the name under which the column positions and the filter selection are saved.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setAutosaveName:` (page 52)

Declared In
`ABPeoplePickerView.h`

clearSearchField:

```
- (void)clearSearchField:(id)sender
```

Discussion
Clears the search field and resets the list of displayed records.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

columnTitleForProperty:

```
- (NSString *)columnTitleForProperty:(NSString *)property
```

Discussion
Returns the title of a custom property.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setColumnTitle:forProperty:` (page 53)

Declared In
`ABPeoplePickerView.h`

deselectAll:

```
- (void)deselectAll:(id)sender
```

Discussion
Deselects all selected groups, records, and values in multi-value properties.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

deselectGroup:

```
- (void)deselectGroup:(ABGroup *)group
```

Discussion
Deselects a group selected in the group list.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectedGroups` (page 49)
`- selectGroup:byExtendingSelection:` (page 50)

Declared In
`ABPeoplePickerView.h`

deselectIdentifier:forPerson:

```
- (void)deselectIdentifier:(NSString *)identifier forPerson:(ABPerson *)person
```

Discussion
Deselects a value selected in a multi-value property.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectedIdentifiersForPerson:` (page 49)
`- selectIdentifier:forPerson:byExtendingSelection:` (page 50)

Declared In
`ABPeoplePickerView.h`

deselectRecord:

```
- (void)deselectRecord:(ABRecord *)record
```

Discussion
Deselects a record selected in the record list.

Availability
Available in Mac OS X v10.3 and later.

See Also
`– selectedRecords` (page 50)
`– selectRecord:byExtendingSelection:` (page 51)

Declared In
`ABPeoplePickerView.h`

displayedProperty

```
- (NSString *)displayedProperty
```

Discussion
Returns the property currently displayed in the record list.

Availability
Available in Mac OS X v10.3 and later.

See Also
`– setDisplayedProperty:` (page 53)

Declared In
`ABPeoplePickerView.h`

editInAddressBook:

```
- (void)editInAddressBook:(id)sender
```

Discussion
Launches Address Book to edit the item selected in the people picker.

Availability
Available in Mac OS X v10.3 and later.

See Also
`– selectInAddressBook:` (page 51)

Declared In
`ABPeoplePickerView.h`

groupDoubleAction

```
- (SEL)groupDoubleAction
```

Discussion
Returns the action invoked when a group is double-clicked.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setGroupDoubleAction:` (page 53)

Declared In
`ABPeoplePickerView.h`

nameDoubleAction

```
- (SEL)nameDoubleAction
```

Discussion
Returns the action invoked when a name is double-clicked.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setNameDoubleAction:` (page 54)

Declared In
`ABPeoplePickerView.h`

properties

```
- (NSArray *)properties
```

Discussion
Returns an array of the properties whose values are shown in the record list.

For additional information about properties see "Person Properties" (page 107)

Availability
Available in Mac OS X v10.3 and later.

See Also
`- addProperty:` (page 44)
`- removeProperty:` (page 49)

Declared In
`ABPeoplePickerView.h`

removeProperty:

`- (void)removeProperty:(NSString *)property`

Discussion
Removes a property from the group of properties whose values are shown in the record list.

For additional information about properties see "Person Properties" (page 107)

Availability
Available in Mac OS X v10.3 and later.

See Also
`- addProperty:` (page 44)
`- properties` (page 48)

Declared In
`ABPeoplePickerView.h`

selectedGroups

`- (NSArray *)selectedGroups`

Discussion
Returns the groups selected in the group list as an array of ABGroup objects.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

selectedIdentifiersForPerson:

`- (NSArray *)selectedIdentifiersForPerson:(ABPerson *)person`

Discussion
Returns the identifiers of the selected values in a multi-value property or `nil` if the property displayed is a single-value property.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectIdentifier:forPerson:byExtendingSelection:` (page 50)
`- deselectIdentifier:forPerson:` (page 46)

Declared In
`ABPeoplePickerView.h`

selectedRecords

```
- (NSArray *)selectedRecords
```

Discussion
Returns the selection in the record list as an array of ABGroup or ABPerson objects.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectRecord:byExtendingSelection:` (page 51)
`- deselectRecord:` (page 47)

Declared In
`ABPeoplePickerView.h`

selectedValues

```
- (NSArray *)selectedValues
```

Discussion
Returns an array of all the values selected in the displayed multi-value property.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

selectGroup:byExtendingSelection:

```
- (void)selectGroup:(ABGroup *)group byExtendingSelection:(BOOL)extend
```

Discussion
Selects a group or a set of groups in the group list.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectedGroups` (page 49)
`- deselectGroup:` (page 46)

Declared In
`ABPeoplePickerView.h`

selectIdentifier:forPerson:byExtendingSelection:

```
- (void)selectIdentifier:(NSString *)identifier forPerson:(ABPerson *)person
    byExtendingSelection:(BOOL)extend
```

Discussion
Selects a value or a set of values in a multi-value property.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- selectedIdentifiersForPerson:` (page 49)
`- deselectIdentifier:forPerson:` (page 46)

Declared In
`ABPeoplePickerView.h`

selectInAddressBook:

```
- (void)selectInAddressBook:(id)sender
```

Discussion
Launches Address Book and selects the item selected in the people picker.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- editInAddressBook:` (page 47)

Declared In
`ABPeoplePickerView.h`

selectRecord:byExtendingSelection:

```
- (void)selectRecord:(ABRecord *)record byExtendingSelection:(BOOL)extend
```

Discussion
Selects a record or a set of records in the record list.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- deselectRecord:` (page 47)
`- selectedRecords` (page 50)

Declared In
`ABPeoplePickerView.h`

setAccessoryView:

```
- (void)setAccessoryView:(NSView *)accessory
```

Discussion
Sets the view that is placed to the left of the search field. If *`accessory`* is `nil`, the accessory view is removed.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- accessoryView` (page 43)

Declared In
`ABPeoplePickerView.h`

setAllowsGroupSelection:

`- (void)setAllowsGroupSelection:(BOOL)flag`

Discussion
Specifies whether the user can select entire groups in the group column (*`flag`* is `YES`). When *`flag`* is `NO`, at least one person in the group is selected when the user selects a group.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- allowsGroupSelection` (page 44)

Declared In
`ABPeoplePickerView.h`

setAllowsMultipleSelection:

`- (void)setAllowsMultipleSelection:(BOOL)flag`

Discussion
Specifies whether multiple groups, records, or values of multi-value properties can be selected at a time.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- allowsMultipleSelection` (page 44)

Declared In
`ABPeoplePickerView.h`

setAutosaveName:

`- (void)setAutosaveName:(NSString *)name`

Discussion
Sets the name under which the column positions and the filter selection are saved.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `autosaveName` (page 45)

Declared In
`ABPeoplePickerView.h`

setColumnTitle:forProperty:

`- (void)setColumnTitle:(NSString *)title forProperty:(NSString *)property`

Discussion
Sets the title for a custom property.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `columnTitleForProperty:` (page 45)

Declared In
`ABPeoplePickerView.h`

setDisplayedProperty:

`- (void)setDisplayedProperty:(NSString *)property`

Discussion
Displays one of the properties whose values are shown in the record list.

Availability
Available in Mac OS X v10.3 and later.

See Also
- `displayedProperty` (page 47)

Declared In
`ABPeoplePickerView.h`

setGroupDoubleAction:

`- (void)setGroupDoubleAction:(SEL)action`

Discussion
Sets the action to be invoked when a group is double-clicked.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- groupDoubleAction` (page 48)

Declared In
`ABPeoplePickerView.h`

setNameDoubleAction:

`- (void)setNameDoubleAction:(SEL)action`

Discussion
Sets the action to be invoked when a name is double-clicked.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- nameDoubleAction` (page 48)

Declared In
`ABPeoplePickerView.h`

setTarget:

`- (void)setTarget:(id)target`

Discussion
Sets the target for double-click actions.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- target` (page 55)

Declared In
`ABPeoplePickerView.h`

setValueSelectionBehavior:

`- (void)setValueSelectionBehavior:(ABPeoplePickerSelectionBehavior)behavior`

Discussion
Specifies the selection behavior, which is `ABSingleValueSelection` by default. Possible values for *behavior* are: `ABNoValueSelection`, `ABSingleValueSelection`, and `ABMultipleValueSelection`.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- valueSelectionBehavior` (page 55)

`ABPeoplePickerSelectionBehavior` (page 103)

Declared In
`ABPeoplePickerView.h`

target

```
- (id)target
```

Discussion
Returns the target of double-click actions.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setTarget:` (page 54)

Declared In
`ABPeoplePickerView.h`

valueSelectionBehavior

```
- (ABPeoplePickerSelectionBehavior)valueSelectionBehavior
```

Discussion
Returns the current selection behavior.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setValueSelectionBehavior:` (page 54)
`ABPeoplePickerSelectionBehavior` (page 103)

Declared In
`ABPeoplePickerView.h`

Constants

These constants are of the type `ABPeoplePickerSelectionBehavior` (page 103) and are used by `setValueSelectionBehavior:` (page 54) and .

Constant	Description
`ABNoValueSelection`	Doesn't allow the user to select individual values. Available in Mac OS X v10.3 and later. Declared in `ABPeoplePickerView.h`.

Constant	Description
`ABSingleValueSelection`	Allows the user to select a single value. Available in Mac OS X v10.3 and later. Declared in `ABPeoplePickerView.h`.
`ABMultipleValueSelection`	Allows the user to select multiple values. Available in Mac OS X v10.3 and later. Declared in `ABPeoplePickerView.h`.

Notifications

ABPeoplePickerGroupSelectionDidChangeNotification

Posted when the selection in the group list is changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

ABPeoplePickerNameSelectionDidChangeNotification

Posted when the selection in the name list is changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

ABPeoplePickerValueSelectionDidChangeNotification

Posted when the selection in a multi-value property is changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

ABPeoplePickerDisplayedPropertyDidChangeNotification

Posted when the displayed property in the record list is changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In

`ABPeoplePickerView.h`

ABPerson Class Reference

Inherits from	ABRecord : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/AddressBook.framework
Availability	Available in Mac OS X v10.2 and later.
Declared in	ABPerson.h
Companion guides	Address Book Programming Guide for Mac OS X Identity Services Programming Guide
Related sample code	ABPresence AddressBookCocoa Birthdays

Overview

The ABPerson class encapsulates all information about a person in the Address Book database—an instance of ABPerson corresponds to a single person record in the database. The ABPerson class defines properties such as the person's name, company, address, email addresses, and phone numbers.

Some of these properties have multiple values that are accessed via standard and user-defined labels. For example, a person may have a home, work, mobile, and fax phone numbers. Therefore, the phone attribute is defined as an ABMultiValue object containing NSString objects for each number. For example, you might get a person's primary phone number as follows:

```
ABMultiValue *phones = [aPerson valueForProperty:kABPhoneProperty];
id value = [phones valueAtIndex:[phones indexForIdentifier:
                [phones primaryIdentifier]]];
```

See ABRecord for common methods to get and set properties. See the "Constants" (page 67) section for a list of all the properties, labels, and keys used to access fields in a person record. See ABMultiValue for more details on multi-value lists and how primary values work.

You can add your own properties to person records too using the `addPropertiesAndTypes:` (page 61) method—that is, attach additional program-defined data to each person record. Because the Address Book database is stored as a property list, these program-defined properties can be ignored by other applications. Note that the AddressBook database is accessed by multiple application and is not encrypted so your application should not store any sensitive information in the database like credit card numbers.

A person may also have an associated picture or image. The image is not actually stored in the Address Book database (a property list)—it's stored in a separate image file. You can set a person's image using the `setImageData:` (page 66) method, or get an image using the `imageData` (page 65) method. Use the NSImage `initWithData:` method to convert an NSData object, returned by the `imageData` method, to an NSImage object.

Image files may be local or remote. Local images are any images in `.../Library/Images/People` or images the user has set using the Address Book application. Remote images are images stored on the network. These images take time to download, so ABPerson provides an asynchronous API for fetching remote images.

Use the `beginLoadingImageDataForClient:` (page 64) method if an image file is not local and you want to perform an asynchronous fetch. You pass a client object that implements the ABImageClient protocol as an argument to this method. The `beginLoadingImageDataForClient:` (page 64) method will return an image tracking number. A `consumeImageData:forTag:` (page 94) message is sent to your client object when the fetch is done. Implement this method to handle the new fetched image. Use the `cancelLoadingImageDataForTag:` (page 62)class method if for some reason you want to cancel an asynchronous fetch.

Person records may belong to multiple groups. Use the `parentGroups` (page 66) method to get the groups a person belongs to. See ABGroup for more information about groups.

You can also search for records matching a particular "query" you specify by creating an ABSearchElement object. Use the `searchElementForProperty:label:key:value:comparison:` (page 63) method to create an ABSearchElement object containing your query. Then use the ABAddressBook `recordsMatchingSearchElement:` (page 18) method, passing the ABSearchElement as the argument, to query the database. See ABSearchElement for more methods that create compound queries.

Your application can also import and export persons in the vCard file format using the `initWithVCardRepresentation:` (page 65)and `vCardRepresentation` (page 66) methods.

The ABPerson class is "toll-free bridged" with its procedural C, opaque type, counterpart. This means that the `ABPersonRef` type is interchangeable in function or method calls with instances of the ABPerson class.

Tasks

Managing Properties

`+ addPropertiesAndTypes:` (page 61)

`+ removeProperties:` (page 62)

`+ properties` (page 62)

`+ typeOfProperty:` (page 64)

Managing Groups

`- parentGroups` (page 66)

Managing Images

`+ cancelLoadingImageDataForTag:` (page 62)

`- beginLoadingImageDataForClient:` (page 64)

`- imageData` (page 65)

`- setImageData:` (page 66)

Searching

`+ searchElementForProperty:label:key:value:comparison:` (page 63)

Importing and Exporting VCard Formatted Files

`- initWithVCardRepresentation:` (page 65)

`- vCardRepresentation` (page 66)

Getting Identity Properties

`- identityUniqueId` (page 65)
Returns the unique identifier for a person.

`- identity` (page 64)
Returns the CSIdentityRef object associated with a person.

Class Methods

addPropertiesAndTypes:

```
+ (NSInteger)addPropertiesAndTypes:(NSDictionary *)properties
```

Discussion
Adds the given properties to all the records of this type in the Address Book database, and returns the number of properties successfully added. In each dictionary entry, the key is a string with the property's name, and the value is a constant with the property's type. The property's name must be unique. You may want to use Java-style package names for your properties, for example, `"org.dogclub.dogname"` or `"com.mycompany.customerID"`. The property type must be one of the constants described in "Constants" (page 78) in ABRecord. Returns `-1` if an error occurs.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABPerson.h`

cancelLoadingImageDataForTag:

```
+ (void)cancelLoadingImageDataForTag:(NSInteger)tag
```

Discussion
Cancels an asynchronous fetch of the images for a given tag. The *`tag`* argument should have been returned from a previous call to the `beginLoadingImageDataForClient:` (page 64) method.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- beginLoadingImageDataForClient:` (page 64)
`- consumeImageData:forTag:` (page 94) (ABImageClient)

Declared In
`ABImageLoading.h`

properties

```
+ (NSArray *)properties
```

Discussion
Returns an array of the names of all the properties for the ABPerson record in the Address Book database.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ typeOfProperty:` (page 64)

Declared In
`ABPerson.h`

removeProperties:

```
+ (NSInteger)removeProperties:(NSArray *)properties
```

Discussion

Removes the given properties from all the records of this type in the Address Book database, and returns the number of properties successfully removed. Returns `-1` if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`ABPerson.h`

searchElementForProperty:label:key:value:comparison:

```
+ (ABSearchElement *)searchElementForProperty:(NSString *)property label:(NSString
    *)label key:(NSString *)key value:(id)value
    comparison:(ABSearchComparison)comparison
```

Discussion

Returns a search element object that specifies a query for records of this type.

- *`property`* is the name of the property to search on, such as `kABAddressProperty` or `kABLastNameProperty`. It cannot be `nil`. For a full list of the properties, see "Constants" (page 67) and "Constants" (page 78) in ABRecord.
- *`label`* is the label name for a multi-value list, such as `kABAddressHomeLabel`, `kABPhoneWorkLabel`, or a user-specified label, such as `"Summer Home"`. If the specified property does not have multiple values, pass `nil`. If the specified property does have multiple values, pass `nil` to search all the values. For a full list of label names, see "Constants" (page 67).
- *`key`* is the key name for a dictionary, such as `kABAddressCityKey` or `kABAddressStreetKey`. If the specified property is not a dictionary, pass `nil`. If the specified property is a dictionary, pass `nil` to search all keys. For a full list of key names, see "Constants" (page 67)
- *`value`* is what you're searching for. If `nil`, then the only supported value for *`comparison`* is `kABEqual` or `kABNotEqual`.
- *`comparison`* specifies the type of comparison to perform and is an `ABSearchComparison` (page 102), such as `kABEqual` or `kABPrefixMatchCaseInsensitive`.

Availability

Available in Mac OS X v10.2 and later.

See Also

`+ searchElementForProperty:label:key:value:comparison:` (page 24) (ABGroup)

`+ searchElementForConjunction:children:` (page 82) (ABSearchElement)

`- recordsMatchingSearchElement:` (page 18) (ABAddressBook)

Related Sample Code

AddressBookCocoa

Birthdays

Declared In

`ABPerson.h`

typeOfProperty:

`+ (ABPropertyType)typeOfProperty:(NSString *)property`

Discussion
Returns the type for a given property. If the property does not exist, this method returns `kABErrorInProperty`.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ properties` (page 62)

Declared In
`ABPerson.h`

Instance Methods

beginLoadingImageDataForClient:

`- (NSInteger)beginLoadingImageDataForClient:(id <ABImageClient>)client`

Discussion
Starts an asynchronous fetch for image data in all locations, and returns a non-zero tag for tracking. The *`client`* object should conform to the ABImageClient protocol. A `consumeImageData:forTag:` (page 94) message is sent to *`client`* when the fetch is done. Use the `cancelLoadingImageDataForTag:` (page 62) method if you need to cancel an asynchronous fetch.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- imageData` (page 65)

Declared In
`ABImageLoading.h`

identity

Returns the CSIdentityRef object associated with a person.

`- (CSIdentityRef)identity`

Return Value
A CSIdentityRef object associated with the receiver or `NULL` if it doesn't exist.

See Also
`- identityUniqueId` (page 65)
`- beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo:`
`- runModalIdentityPicker`

identityUniqueId

Returns the unique identifier for a person.

```
- (NSString*)identityUniqueId
```

Return Value
A unique identifier for the receiver or `nil` if it doesn't exist.

See Also
- `identity` (page 64)
- `beginSheetModalForWindow:modalDelegate:didEndSelector:contextInfo:`
- `runModalIdentityPicker`

imageData

```
- (NSData *)imageData
```

Discussion
Returns data that contains a picture of this person. This method searches only the local file system and operates synchronously. To perform an asynchronous search or to search over a network, use `beginLoadingImageDataForClient:` (page 64).

The returned data is in a QuickTime-compatible format. To create an image from it, use the NSImage method `initWithData:`.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `beginLoadingImageDataForClient:` (page 64)
- `setImageData:` (page 66)

Declared In
`ABImageLoading.h`

initWithVCardRepresentation:

```
- (id)initWithVCardRepresentation:(NSData *)vCardData
```

Discussion
Returns an ABPerson instance initialized with the given data. If *`vCardData`* is `nil` or is not a valid vCard format, this method returns `nil`.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `vCardRepresentation` (page 66)

Declared In
`ABPerson.h`

parentGroups

`- (NSArray *)parentGroups`

Discussion
Returns an array of the ABGroups this person belongs to. If the person doesn't belong to any groups, this method returns an empty array.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABPerson.h`

setImageData:

`- (BOOL)setImageData:(NSData *)data`

Discussion
Sets the image for this person to the given data. The *`data`* argument must be in a QuickTime-compatible format. Pass `nil` to specify that there is no image for this person.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- imageData` (page 65)
`- beginLoadingImageDataForClient:` (page 64)

Declared In
`ABImageLoading.h`

vCardRepresentation

`- (NSData *)vCardRepresentation`

Discussion
Returns the vCard representation of the person as a data object in vCard format.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- initWithVCardRepresentation:` (page 65)

Declared In
`ABPerson.h`

Constants

These are the properties that an ABPerson record contains by default. ABRecord describes additional properties that all records contain, in "Constants" (page 78). Note that some of these properties are not displayed in the Address Book application. Developers can add their own properties with the ABRecord method `addPropertiesAndTypes:` (page 61).

Note: The `kABHomePageProperty` property has been deprecated in Mac OS X version 10.4. Instead, use the `kABURLsProperty` multi-value property.

Constant	Description
`kABFirstNameProperty`	First name (NSString)
`kABLastNameProperty`	Last name (NSString)
`kABFirstNamePhoneticProperty`	Phonetic representation of the first name (NSString)
`kABLastNamePhoneticProperty`	Phonetic representation of the last name (NSString)
`kABBirthdayProperty`	Birth date (NSDate)
`kABOrganizationProperty`	Company name (NSString)
`kABJobTitleProperty`	Job title (NSString)
`kABHomePageProperty`	Home web page (NSString)
`kABURLsProperty`	Web pages(ABMultiValue containing NSString)
`kABCalendarURIsProperty`	Calendar URIs(ABMultiValue containing NSString)
`kABEmailProperty`	Email addresses (ABMultiValue containing NSString)
`kABAddressProperty`	Street addresses (ABMultiValue containing NSDictionary)
`kABPhoneProperty`	Generic phone number (ABMultiValue containing NSString)
`kABAIMInstantProperty`	AOL instant messaging ID (ABMultiValue containing NSString)
`kABJabberInstantProperty`	Jabber instant messaging ID (ABMultiValue containing NSString)
`kABMSNInstantProperty`	MSN instant messaging ID (ABMultiValue containing NSString)
`kABYahooInstantProperty`	Yahoo instant messaging ID (ABMultiValue containing NSString)
`kABICQInstantProperty`	ICQ instant messaging ID (ABMultiValue containing NSString)
`kABNoteProperty`	Notes. (NSString)
`kABMiddleNameProperty`	Middle name. This property isn't displayed in the Address Book application. (NSString)

Constant	Description
`kABMiddleNamePhoneticProperty`	Phonetic representation of the middle name. This property isn't displayed in the Address Book application. (NSString)
`kABTitleProperty`	Title, such as "Mr.," "Mrs.," "General," "Cardinal," or "Lord." This property isn't displayed in the Address Book application. (NSString)
`kABSuffixProperty`	Suffix, such as "Sr.," "Jr.," "III.," or "Esq." This property isn't displayed in the Address Book application. (NSString)
`kABNicknameProperty`	Nickname. This property isn't displayed in the Address Book application. (NSString)
`kABMaidenNameProperty`	Maiden name. This property isn't displayed in the Address Book application. (NSString)
`kABOtherDatesProperty`	Dates associated with a person (ABMultiDateProperty containing dates).
`kABRelatedNamesProperty`	Names of people related to a person (ABMultiStringProperty containing names).
`kABDepartmentProperty`	Department name.
`kABPersonFlags`	Property that specifies the name ordering and configuration of a record in the Address Book application.

The ABPersonFlags property is used to access the following settings:

Constant	Description
`kABShowAsPerson`	Record is displayed as a person.
`kABShowAsCompany`	Record is displayed as a company.
`kABShowAsMask`	Used in conjuction with `kABShowAsPerson` and `kABShowAsCompany` to determine record configuration.
`kABDefaultNameOrdering`	Default name ordering (whether a person's first name or last name is displayed first) the Address Book application.
`kABFirstNameFirst`	First name is displayed first in Address Book.
`kABLastNameFirst`	Last name is displayed first. in Address Book.
`kABNameOrderingMask`	Used in conjunction with `kABDefaultNameOrdering`, `kABFirstNameFirst`, and `kABLastNameFirst` to determine name ordering.

These are the default labels contained in the Address Book database for specifying different values in a multi-value list. Users can also add their own labels.

Constant	Description
`kABEmailWorkLabel`	Home email
`kABEmailHomeLabel`	Work email
`kABAddressHomeLabel`	Home address
`kABAddressWorkLabel`	Work address
`kABPhoneWorkLabel`	Work phone number
`kABPhoneHomeLabel`	Home phone number
`kABPhoneMobileLabel`	Cell phone number
`kABPhoneMainLabel`	Main phone number
`kABPhoneHomeFAXLabel`	Home FAX number
`kABPhoneWorkFAXLabel`	Work FAX number
`kABPhonePagerLabel`	Pager number
`kABHomePageLabel`	Web page URL
`kABAIMWorkLabel`	Work AOL instant messaging ID
`kABAIMHomeLabel`	Home AOL instant messaging ID
`kABJabberWorkLabel`	Work Jabber instant messaging ID
`kABJabberHomeLabel`	Home Jabber instant messaging ID
`kABMSNWorkLabel`	Work MSN instant messaging ID
`kABMSNHomeLabel`	Home MSN instant messaging ID
`kABYahooWorkLabel`	Work Yahoo instant messaging ID
`kABYahooHomeLabel`	Home Yahoo instant messaging ID
`kABICQWorkLabel`	Work ICQ instant messaging ID
`kABICQHomeLabel`	Home ICQ instant messaging ID
`kABAnniversaryLabel`	Anniversary date
`kABMotherLabel`	Mother
`kABFatherLabel`	Father
`kABParentLabel`	Parent
`kABSisterLabel`	Sister
`kABBrotherLabel`	Brother

Constant	Description
`kABChildLabel`	Child
`kABFriendLabel`	Friend
`kABSpouseLabel`	Spouse
`kABPartnerLabel`	Partner
`kABAssistantLabel`	Assistant
`kABManagerLabel`	Manager

Use these labels in searches to match all work, home, or other labels. For example, `kABWorkLabel` can be used in place of `kABEmailWorkLabel` or `kABAddressWorkLabel`.

Constant	Description
`kABWorkLabel`	All Work labels match this label
`kABHomeLabel`	All Home labels match this label
`kABOtherLabel`	All labels other than Home or Work labels match this label.

These are the keys contained in each address dictionary. Neither developers nor users can add more keys.

Constant	Description
`kABAddressStreetKey`	Street
`kABAddressCityKey`	City
`kABAddressStateKey`	State
`kABAddressZIPKey`	Zip
`kABAddressCountryKey`	Country
`kABAddressCountryCodeKey`	Country Code

The `kABAddressCountryCodeKey` must be one of the following ISO country codes.

Country Codes	Description
ae	United Arab Emirates
ar	Argentina
at	Austria
au	Australia

Country Codes	Description
ba	Bosnia and Herzegovina
be	Belgium
bh	Bahrain
br	Brazil
ca	Canada
ch	Switzerland
cn	China
cz	Czech Republic
de	Germany
dk	Denmark
eg	Egypt
es	Spain
fi	Finland
fr	France
gl	Greenland
gr	Greece
hk	Hong Kong
hr	Croatia
hu	Hungary
ie	Ireland
il	Israel
id	Indonesia
in	India
is	Iceland
it	Italy
jp	Japan
jo	Jordan

Country Codes	Description
kr	South Korea
kw	Kuwait
lb	Lebanon
lu	Luxembourg
mk	Macedonia
mx	Mexico
nl	Netherlands
no	Norway
nz	New Zealand
om	Oman
pl	Poland
pt	Portugal
qa	Qatar
ro	Romania
ru	Russian Federation
sa	Saudi Arabia
se	Sweden
sg	Singapore
si	Slovenia
sk	Slovakia
sy	Syrian Arab Republic
tr	Turkey
tw	Taiwan
ua	Ukraine
gb	United Kingdom
us	United States
ye	Yemen

Country Codes	Description
yu	Serbia and Montenegro
za	South Africa

ABRecord Class Objective-C Reference

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

Overview

ABRecord is an abstract superclass providing a common interface to and defining common properties for all Address Book records. A property is a field in the database record such as the first or last name of a person record. ABRecord defines the types of properties supported, and basic methods for getting, setting, and removing property values.

Use `valueForProperty:` (page 78) to get a record's property value, use `setValue:forProperty:` (page 77) to set a value, and `removeValueForProperty:` (page 77) to remove a value. Don't just invoke `setValue:forProperty:` with `nil` as the property value argument, it will raises an exception.

Each record in the Address Book database has a corresponding unique ID obtained using the `uniqueId` (page 78) method. The unique ID is used by other classes in the AddressBook framework.

Use `isReadOnly` (page 77) to determine whether or not a record is read-only.

The ABRecord class is "toll-free bridged" with its procedural C, opaque type, counterpart. This means that the `ABRecordRef` type is interchangeable in function or method calls with instances of the ABRecord class.

Tasks

Initializing

- `initWithAddressBook:` (page 76)
  Initializes the receiver and adds it to the given address book.

Retrieving and Setting Values

- `removeValueForProperty:` (page 77)

- `setValue:forProperty:` (page 77)

- `valueForProperty:` (page 78)

Retrieving a Specific Record

- `isReadOnly` (page 77)

Determining Properties

- `uniqueId` (page 78)

Instance Methods

initWithAddressBook:

Initializes the receiver and adds it to the given address book.

```
- (id)initWithAddressBook:(ABAddressBook *)addressBook
```

Parameters

addressBook
  An address book that you want to initialize the receiver with.

Discussion

The receiver is added to *addressBook* but is not visible to other address books until *addressBook* is saved. This is the designated initializer for this class.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`ABRecord.h`

isReadOnly

```
- (BOOL)isReadOnly
```

Discussion
Returns YES if the record is read-only, NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`ABRecord.h`

removeValueForProperty:

```
- (BOOL)removeValueForProperty:(NSString *)property
```

Discussion
Removes the value for a given property. When you next call `valueForProperty:` (page 78) on that property, it returns `nil`.

If property is `nil`, this method raises an exception. This method returns `YES` if the value is removed successfully, `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABRecord.h`

setValue:forProperty:

```
- (BOOL)setValue:(id)value forProperty:(NSString *)property
```

Discussion
Sets the value of a given property for a record. The type of the value must match the property's type (see "Constants" (page 78) for a list of possible property types). If *`property`* is `nil` or if *`value`* is not of the correct type, this method raises an exception. If *`property`* is a multi-value list property, this method checks to see if the values in the multi-value list are the same type. If the multi-value list contains mixed types, this method returns `NO`. This method returns `YES` if the value was set successfully, and `NO` otherwise.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABRecord.h`

uniqueId

```
- (NSString *)uniqueId
```

Discussion
Returns the unique ID of the receiver. This method is equivalent to invoking `valueForProperty:` (page 78) passing `kABUIDProperty` as the argument.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- recordForUniqueId:` (page 17) (ABAddressBook)

Declared In
`ABRecord.h`

valueForProperty:

```
- (id)valueForProperty:(NSString *)property
```

Discussion
Returns the value of the given property. The type of the value depends on the property type (see "Constants" (page 78) for a list of possible property types). Note that the retuned value is always of an immutable type (for example, an NSString not an NSMutableString is returned).

If *`property`* is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
AddressBookCocoa

Declared In
`ABRecord.h`

Constants

These are of type `ABPropertyType` (page 101) and describe the possible types for ABRecord properties:

Constant	Description
`kABStringProperty`	Indicates a NSString object. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.

Constant	Description
`kABIntegerProperty`	Indicates a NSNumber object representing an integer. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABRealProperty`	Indicates a NSNumber object representing a real number. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABDateProperty`	Indicates a NSDate object. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABArrayProperty`	Indicates a NSArray object. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABDictionaryProperty`	Indicates a NSDictionary object. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABDataProperty`	Indicates a NSData object. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABMultiStringProperty`	Indicates a ABMultiValue containing NSString objects. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABMultiIntegerProperty`	Indicates a ABMultiValue containing NSNumber objects representing integers. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABMultiRealProperty`	Indicates a ABMultiValue containing NSNumber objects representing real numbers. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABMultiDateProperty`	Indicates a ABMultiValue containing NSDate objects. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABMultiArrayProperty`	Indicates a ABMultiValue containing NSArray objects. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.

Constant	Description
`kABMultiDictionary-Property`	Indicates a ABMultiValue containing NSDictionary objects. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABMultiDataProperty`	Indicates a ABMultiValue containing NSData objects. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABErrorInProperty`	Returned by some methods when an invalid property is used. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.

These properties are in all types of records.

Constant	Description
`kABUIDProperty`	The unique ID for this record. It's guaranteed never to change, no matter how much the record changes. If you need to store a reference to an ABRecord, use this value. (NSString)
`kABCreationDateProperty`	The date when the record was first saved (NSDate)
`kABModificationDateProperty`	The date when the record was last saved (NSDate)

ABSearchElement Class Objective-C Reference

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

Overview

ABSearchElement is used to specify a search query for records in the Address Book database.

You can create a simple query by creating an ABSearchElement object using either the ABGroup `searchElementForProperty:label:key:value:comparison:` (page 24) or ABPerson `searchElementForProperty:label:key:value:comparison:` (page 63) methods. Then you use the ABAddressBook `recordsMatchingSearchElement:` (page 18) method, passing the ABSearchElement as the argument, to query the database.

ABSearchElement also provides a method for creating compound queries. Use the `searchElementForConjunction:children:` (page 82) method to combine two simple or complex queries into a compound query using either the `kABSearchAnd` or `kABSearchOr` conjunction constants.

Use the `matchesRecord:` (page 82) function to test whether a specific record matches a query.

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

Tasks

Searching

`+ searchElementForConjunction:children:` (page 82)

Matching

`- matchesRecord:` (page 82)

Class Methods

searchElementForConjunction:children:

```
+ (ABSearchElement *)searchElementForConjunction:(ABSearchConjunction)conjunction
    children:(NSArray *)children
```

Discussion
Returns a compound search element created by combining the search elements in an array with the given conjunction. The objects in the *`children`* array must be ABSearchElement objects. The conjunction can be `kABSearchAnd` or `kABSearchOr`. If *`children`* is `nil` or empty, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ searchElementForProperty:label:key:value:comparison:` (page 63) (ABPerson)
`+ searchElementForProperty:label:key:value:comparison:` (page 24) (ABGroup)
`- recordsMatchingSearchElement:` (page 18) (ABAddressBook)

Declared In
`ABSearchElement.h`

Instance Methods

matchesRecord:

```
- (BOOL)matchesRecord:(ABRecord *)record
```

Discussion
Tests whether or not a record matches a search element. Returns `YES` if the *`record`* argument satisfies the conditions in the search element, `NO` otherwise. If *`record`* is `nil`, this method raises an exception.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ searchElementForProperty:label:key:value:comparison:` (page 63) (ABPerson)
`+ searchElementForProperty:label:key:value:comparison:` (page 24) (ABGroup)
`+ searchElementForConjunction:children:` (page 82)

Declared In
`ABSearchElement.h`

Constants

These constants are of the type `ABSearchConjunction` (page 102) and are used by `searchElementForConjunction:children:` (page 82).

Constant	Description
`kABSearchAnd`	Join the search elements together with the AND operand. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABSearchOr`	Join the search elements together with the OR operand. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.

These constants are of the type `ABSearchComparison` (page 102) and are used by the ABPerson method `searchElementForProperty:label:key:value:comparison:` (page 63) and the ABGroup method `searchElementForProperty:label:key:value:comparison:` (page 24).

Constant	Description
`kABEqual`	Search for elements that are equal to the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABNotEqual`	Search for elements that are not equal to the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABNotEqualCase-Insensitive`	Search for elements that are not equal to the value, ignoring case. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.

Constant	Description
`kABLessThan`	Search for elements that are less than the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABLessThanOrEqual`	Search for elements that are less than or equal to the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABGreaterThan`	Search for elements that are greater than the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABGreaterThanOrEqual`	Search for elements that are greater than or equal to the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABEqualCaseInsensitive`	Search for elements that are equal to the value, ignoring case. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABContainsSubString`	Search for elements that contain the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABContainsSubString-CaseInsensitive`	Search for elements that contain the value, ignoring case. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABPrefixMatch`	Search for elements that begin with the value. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABPrefixMatchCase-Insensitive`	Search for elements that begin with the value, ignoring case. Available in Mac OS X v10.2 and later. Declared in `ABTypedefs.h`.
`kABSuffixMatch`	Search for elements that end with the value. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABSuffixMatchCase-Insensitive`	Search for elements that end with the value, ignoring case. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.

Constant	Description
`kABBitsInBitFieldMatch`	Search for elements that match the bits in ABPersonFlags. (Available in Mac OS X v10.3 and later.) Available in Mac OS X v10.3 and later. Declared in `ABTypedefs.h`.
`kABDoesNotContain-SubString`	Search for elements that do not contain the value. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABDoesNotContain-SubStringCaseInsensitive`	Search for elements that do not contain the value, ignoring case. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABWithinInterval-AroundToday`	Search for elements that are within a time interval (in seconds) forward or backward from today. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABWithinInterval-AroundTodayYearless`	Search for elements that are within a time interval (in seconds) forward or backward from this day in any year. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABNotWithinInterval-AroundToday`	Search for elements that are *not* within a time interval (in seconds) forward or backward from today. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABNotWithinInterval-AroundTodayYearless`	Search for elements that are *not* within a time interval (in seconds) forward or backward from this day in any year. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABWithinInterval-FromToday`	Search for elements that are within a time interval (in seconds) forward from today. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABWithinInterval-FromTodayYearless`	Search for elements that are within a time interval (in seconds) forward from this day in any year. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.

Constant	Description
`kABNotWithinInterval-FromToday`	Search for elements that are *not* within a time interval (in seconds) forward from today. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.
`kABNotWithinInterval-FromTodayYearless`	Search for elements that are *not* within a time interval (in seconds) forward from this day in any year. (Available in Mac OS X v10.4 and later.) Available in Mac OS X v10.4 and later. Declared in `ABTypedefs.h`.

Protocols

ABActionDelegate Protocol Objective-C Reference

~~(informal protocol)~~

Framework	/System/Library/Frameworks/AddressBook.framework
Declared in	ABActions.h
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Address Book Programming Guide for Mac OS X

Overview

The ABActionDelegate informal protocol allows you to populate the rollover menus of Address Book with custom items. You do this by implementing an Address Book action plug-in. The plug-in's NSBundle must implement `actionProperty:`, `titleForPerson:identifier:` and `performActionForPerson:identifier:`.

Each action plug-in can implement only one action. Actions can only apply to items with labels.

Use Xcode to create Address Book action plug-ins. Place action plug-ins in `~/Library/Address Book Plug-Ins` or `/Library/Address Book Plug-Ins`, depending on the scope you want for the action.

Tasks

Performing Actions

– `performActionForPerson:identifier:` (page 90)

Queries

– `actionProperty` (page 90)

– `shouldEnableActionForPerson:identifier:` (page 90)

– `titleForPerson:identifier:` (page 91)

Instance Methods

actionProperty

```
- (NSString *)actionProperty
```

Discussion
Sent to the delegate to request the ABProperty ("Person Properties" (page 107)) the action applies to.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABActions.h`

performActionForPerson:identifier:

```
- (void)performActionForPerson:(ABPerson *)person identifier:(NSString *)identifier
```

Discussion
Sent to the delegate to perform the action. If the property returned by `actionProperty` (page 90) is a multi-value property, *`identifier`* contains the unique identifier of the value selected.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABActions.h`

shouldEnableActionForPerson:identifier:

```
- (BOOL)shouldEnableActionForPerson:(ABPerson *)person identifier:(NSString
    *)identifier
```

Discussion
Sent to the delegate to determine whether the action should be enabled. If the property returned by `actionProperty` (page 90) is a multi-value property, *`identifier`* contains the unique identifier of the value selected.

Return `YES` is the action is applicable and `NO` otherwise.

Availability
Available in Mac OS X v10.3 or later.

Declared In
`ABActions.h`

titleForPerson:identifier:

```
- (NSString *)titleForPerson:(ABPerson *)person identifier:(NSString *)identifier
```

Discussion
Sent to the delegate to request the title of the menu item for the action. If the property returned by `actionProperty` (page 90) is a multi-value property, *`identifier`* contains the unique identifier of the value selected.

Return the title of the menu item for the action.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABActions.h`

ABImageClient Protocol Objective-C Reference

Adopted by	ABPerson
Conforms to	NSObject
Framework	/System/Library/Frameworks/AddressBook.framework
Availability	Available in Mac OS X v10.2 and later.
Declared in	ABImageLoading.h
Companion guide	Address Book Programming Guide for Mac OS X

Overview

Implement this protocol to handle images loaded from an asynchronous fetch for ABPerson objects.

A person may have an associated picture or image. The image in not actually stored in the Address Book database (a property list)—it's stored in a separate image file. These image files may be local or remote. Local images are any images in `.../Library/Images/People` or images the user has set using the Address Book application. Remote images are images stored on the network. Theses images take time to download, so ABPerson provides an asynchronous API for fetching remote images.

Use the `beginLoadingImageDataForClient:` (page 64) method if an image file is not local and you want to perform an asynchronous fetch. You pass a client object that implements the ABImageClient protocol as an argument to this method. The `beginLoadingImageDataForClient:` (page 64) method will return an image tracking number. A `consumeImageData:forTag:` (page 94) message is sent to your client object when the fetch is done. Implement this method to handle the new fetched image. Use the `cancelLoadingImageDataForTag:` (page 62)class method if for some reason you want to cancel an asynchronous fetch.

Tasks

Loading an Image

– `consumeImageData:forTag:` (page 94)

Instance Methods

consumeImageData:forTag:

```
- (void)consumeImageData:(NSData *)data forTag:(NSInteger)tag
```

Discussion
Gets the image data for the given tag that was initiated by an asynchronous fetch. The *data* argument is set to an NSImage/QuickTime compatible format or `nil` if no image could be found. The *tag* argument should have been obtained from a previous call to the ABPerson `beginLoadingImageDataForClient:` (page 64) method. In the case of a multi-threaded application, this method is always called on the main thread.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ cancelLoadingImageDataForTag:` (page 62) (ABPerson)
`- imageData` (page 65) (ABPerson)

Declared In
`ABImageLoading.h`

Functions

AddressBook Functions Reference

Framework:	AddressBook/AddressBook.h

Overview

This chapter describes the functions and function-like macros found in AddressBook.

Functions

ABLocalizedPropertyOrLabel

Returns the localized version of a built in property, label, or key.

```
NSString * ABLocalizedPropertyOrLabel (
   NSString *propertyOrLabel
);
```

Discussion
The *propertyOrLabel* argument is the property, label, or key you wish to localize. Returns *propertyOrLabel* if a localized string can not be found

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABGlobals.h`

Data Types

Address Book Data Types Reference

Framework: AddressBook/AddressBook.h

Overview

This chapter describes the data types and constants found in the Address Book framework.

Data Types

ABPropertyType

Defines the possible types of ABRecord properties.

```
enum _ABPropertyType {
    kABErrorInProperty            = 0x0,
    kABStringProperty             = 0x1,
    kABIntegerProperty            = 0x2,
    kABRealProperty               = 0x3,
    kABDateProperty               = 0x4,
    kABArrayProperty              = 0x5,
    kABDictionaryProperty         = 0x6,
    kABDataProperty               = 0x7,
    kABMultiStringProperty        = kABMultiValueMask | kABStringProperty,
    kABMultiIntegerProperty       = kABMultiValueMask | kABIntegerProperty,
    kABMultiRealProperty          = kABMultiValueMask | kABRealProperty,
    kABMultiDateProperty          = kABMultiValueMask | kABDateProperty,
    kABMultiArrayProperty         = kABMultiValueMask | kABArrayProperty,
    kABMultiDictionaryProperty    = kABMultiValueMask | kABDictionaryProperty,
    kABMultiDataProperty          = kABMultiValueMask | kABDataProperty
};
typedef CFIndex ABPropertyType;
```

Discussion
These constants are described in "Constants" (page 78) in "ABRecord".

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABTypedefs.h`

ABSearchComparison

Defines constants used to construct search elements.

```
enum _ABSearchComparison {
        kABEqual,
        kABNotEqual,
        kABLessThan,
        kABLessThanOrEqual,
        kABGreaterThan,
        kABGreaterThanOrEqual,
        kABEqualCaseInsensitive,
        kABContainsSubString,
        kABContainsSubStringCaseInsensitive,
        kABPrefixMatch,
        kABPrefixMatchCaseInsensitive,
        kABBitsInBitFieldMatch,
        kABDoesNotContainSubString,
        kABDoesNotContainSubStringCaseInsensitive,
        kABNotEqualCaseInsensitive,
        kABSuffixMatch,
        kABSuffixMatchCaseInsensitive,
        kABWithinIntervalAroundToday,
        kABWithinIntervalAroundTodayYearless,
        kABNotWithinIntervalAroundToday,
        kABNotWithinIntervalAroundTodayYearless,
        kABWithinIntervalFromToday,
        kABWithinIntervalFromTodayYearless,
        kABNotWithinIntervalFromToday,
        kABNotWithinIntervalFromTodayYearless
};
typedef CFIndex ABSearchComparison;
```

Discussion
These constants are described in "Constants" (page 83) in "ABSearchElement".

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABTypedefs.h`

ABSearchConjunction

Defines constants used to combine search elements.

```
enum _ABSearchConjunction {
        kABSearchAnd,
        kABSearchOr
};
typedef CFIndex ABSearchConjunction;
```

Discussion
These constants are described in "Constants" (page 83) in "ABSearchElement".

Availability
Available in Mac OS X v10.2 and later.

Declared In
`ABTypedefs.h`

ABPeoplePickerSelectionBehavior

```
typedef enum {
    ABNoValueSelection   = 0,
    ABSingleValueSelection   = 1,
    ABMultipleValueSelection = 2
} ABPeoplePickerSelectionBehavior;
```

Discussion
These constants are described in "Constants" (page 55) in "ABPeoplePickerView".

Availability
Available in Mac OS X v10.3 and later.

Declared In
`ABPeoplePickerView.h`

Constants

Address Book Constants Reference

Framework: AddressBook/AddressBook.h

Overview

Constants

Global Variables

Record Properties

Properties common to all types of records.

```
extern NSString * const kABUIDProperty;
extern NSString * const kABCreationDateProperty;
extern NSString * const kABModificationDateProperty;
```

Discussion
These constants are described in "Constants" (page 78) in "ABRecord".

Person Properties

Properties common to all persons.

```
extern NSString * const kABFirstNameProperty;
extern NSString * const kABLastNameProperty;
extern NSString * const kABFirstNamePhoneticProperty;
extern NSString * const kABLastNamePhoneticProperty;
extern NSString * const kABBirthdayProperty;
extern NSString * const kABOrganizationProperty;
extern NSString * const kABJobTitleProperty;
extern NSString * const kABHomePageProperty;
extern NSString * const kABURLsProperty;
extern NSString * const kABCalendarURIsProperty;
extern NSString * const kABEmailProperty;
extern NSString * const kABAddressProperty;
extern NSString * const kABPhoneProperty;
extern NSString * const kABAIMInstantProperty;
extern NSString * const kABJabberInstantProperty;
extern NSString * const kABMSNInstantProperty;
extern NSString * const kABYahooInstantProperty;
extern NSString * const kABICQInstantProperty;
extern NSString * const kABNoteProperty;
extern NSString * const kABMiddleNameProperty;
extern NSString * const kABMiddleNamePhoneticProperty;
extern NSString * const kABTitleProperty;
extern NSString * const kABSuffixProperty;
extern NSString * const kABNicknameProperty;
extern NSString * const kABMaidenNameProperty;
extern NSString * const kABOtherDatesProperty;
extern NSString * const kABRelatedNamesProperty;
extern NSString * const kABDepartmentProperty;
extern NSString * const kABPersonFlags;
```

Discussion
These constants are described in “Constants” (page 67) in “ABPerson”.

Address Keys

Keys used to specify the different fields in a `kABAddressProperty` for person records.

```
extern NSString * const kABAddressStreetKey;
extern NSString * const kABAddressCityKey;
extern NSString * const kABAddressStateKey;
extern NSString * const kABAddressZIPKey;
extern NSString * const kABAddressCountryKey;
extern NSString * const kABAddressCountryCodeKey;
```

Discussion
These constants as well as the possible values of `kABAddressCountryCodeKey` are described in “Constants” (page 67) in “ABPerson”.

Multi-value List Labels

Pre-defined labels used by a multi-value list.

```
extern NSString * const kABEmailWorkLabel;
extern NSString * const kABEmailHomeLabel;
extern NSString * const kABAddressHomeLabel;
extern NSString * const kABAddressWorkLabel;
extern NSString * const kABPhoneWorkLabel;
extern NSString * const kABPhoneHomeLabel;
extern NSString * const kABPhoneMobileLabel;
extern NSString * const kABPhoneMainLabel;
extern NSString * const kABPhoneHomeFAXLabel;
extern NSString * const kABPhoneWorkFAXLabel;
extern NSString * const kABPhonePagerLabel;
extern NSString * const kABAIMWorkLabel;
extern NSString * const kABAIMHomeLabel;
extern NSString * const kABJabberWorkLabel;
extern NSString * const kABJabberHomeLabel;
extern NSString * const kABMSNWorkLabel;
extern NSString * const kABMSNHomeLabel;
extern NSString * const kABYahooWorkLabel;
extern NSString * const kABYahooHomeLabel;
extern NSString * const kABICQWorkLabel;
extern NSString * const kABICQHomeLabel;
extern NSString * const kABAnniversaryLabel;
extern NSString * const kABMotherLabel;
extern NSString * const kABFatherLabel;
extern NSString * const kABParentLabel;
extern NSString * const kABSisterLabel;
extern NSString * const kABBrotherLabel;
extern NSString * const kABChildLabel;
extern NSString * const kABFriendLabel;
extern NSString * const kABSpouseLabel;
extern NSString * const kABPartnerLabel;
extern NSString * const kABAssistantLabel;
extern NSString * const kABManagerLabel;
extern NSString * const kABHomePageProperty;
```

Discussion
These constants are described in "Constants" (page 67) in "ABPerson".

Generic Labels

Labels that can be used to match all work, home, or other labels.

```
extern NSString * const kABWorkLabel;
extern NSString * const kABHomeLabel;
extern NSString * const kABOtherLabel;
```

Discussion
These constants are described in "Constants" (page 67) in "ABPerson".

Constants

kABMultiValueMask

Indicates a multi-value property type.

```
#define kABMultiValueMask 0x100
```

Discussion
Used by `ABPropertyType` (page 101) to specify a multi-value property.

Person Flags

The ABPersonFlags property ("Person Properties" (page 107)) is used in conjunction with the following symbols:

```
#define kABShowAsPerson          000000
#define kABShowAsCompany         000001
#define kABShowAsMask            000007
#define kABDefaultNameOrdering   000000
#define kABFirstNameFirst        000040
#define kABLastNameFirst         000020
#define kABNameOrderingMask      000070
```

Discussion
These symbols are explained in the "Constants" section of ABPerson.

Notifications

kABDatabaseChangedNotification

Notification sent when the Address Book database has changed.

```
extern NSString * const kABDatabaseChangedNotification;
```

Discussion
This notification is described in the "Notifications" (page 20) section of ABAddressBook.

kABDatabaseChangedExternallyNotification

Notification sent when the Address Book database changes from an external application.

```
extern NSString * const kABDatabaseChangedExternallyNotification;
```

Discussion
This notification is described in the "Notifications" (page 20) section of ABAddressBook.

Document Revision History

This table describes the changes to *Address Book Objective-C Framework Reference*.

Date	Notes
2007-07-08	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a collection of separate documents.

Index