
NSKeyedUnarchiver Class Reference

[Cocoa](#) > [Data Management](#)



2008-10-15



Apple Inc.
© 2008 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, Leopard, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY

DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSKeyedUnarchiver Class Reference 5

Overview	5
Tasks	6
Initializing a Keyed Unarchiver	6
Unarchiving Data	6
Decoding Data	6
Managing the Delegate	7
Managing Class Names	7
Decoding Objects	7
Finishing Decoding	7
Class Methods	7
classForClassName:	7
setClass:forClassName:	8
unarchiveObjectWithData:	8
unarchiveObjectWithFile:	9
Instance Methods	10
classForClassName:	10
containsValueForKey:	10
decodeBoolForKey:	11
decodeBytesForKey:returnedLength:	11
decodeDoubleForKey:	12
decodeFloatForKey:	12
decodeInt32ForKey:	13
decodeInt64ForKey:	13
decodeIntForKey:	13
decodeObjectForKey:	14
delegate	14
finishDecoding	15
initWithReadingWithData:	15
setClass:forClassName:	16
setDelegate:	16
Delegate Methods	16
unarchiver:cannotDecodeObjectOfClassName:originalClasses:	16
unarchiver:didDecodeObject:	17
unarchiver:willReplaceObject:withObject:	18
unarchiverDidFinish:	18
unarchiverWillFinish:	18
Constants	19
Keyed Unarchiving Exception Names	19

Document Revision History 21

Index 23

NSKeyedUnarchiver Class Reference

Inherits from	NSCoder : NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.2 and later.
Companion guide	Archives and Serializations Programming Guide for Cocoa
Declared in	NSKeyedArchiver.h
Related sample code	CoreRecipes CustomAtomicStoreSubclass iSpend QTQuartzPlayer Squiggles

Overview

`NSKeyedUnarchiver`, a concrete subclass of `NSCoder`, defines methods for decoding a set of named objects (and scalar values) from a keyed archive. Such archives are produced by instances of the `NSKeyedArchiver` class.

A keyed archive is encoded as a hierarchy of objects. Each object in the hierarchy serves as a namespace into which other objects are encoded. The objects available for decoding are restricted to those that were encoded within the immediate scope of a particular object. Objects encoded elsewhere in the hierarchy, whether higher than, lower than, or parallel to this particular object, are not accessible. In this way, the keys used by a particular object to encode its instance variables need to be unique only within the scope of that object.

If you invoke one of the `decode...` methods of this class using a key that does not exist in the archive, a non-positive value is returned. This value varies by decoded type. For example, if a key does not exist in an archive, `decodeBoolForKey:` (page 11) returns `NO`, `decodeIntForKey:` (page 13) returns `0`, and `decodeObjectForKey:` (page 14) returns `nil`.

`NSKeyedUnarchiver` supports limited type coercion. A value encoded as any type of integer, whether a standard `int` or an explicit 32-bit or 64-bit integer, can be decoded using any of the integer decode methods. Likewise, a value encoded as a `float` or `double` can be decoded as either a `float` or a `double` value. If an encoded value is too large to fit within the coerced type, the decoding method raises an `NSRangeException`. Further, when trying to coerce a value to an incompatible type, for example decoding an `int` as a `float`, the decoding method raises an `NSInvalidUnarchiveOperationException`.

Tasks

Initializing a Keyed Unarchiver

- [initWithReadingWithData:](#) (page 15)
Initializes the receiver for decoding an archive previously encoded by `NSKeyedArchiver`.

Unarchiving Data

- + [unarchiveObjectWithData:](#) (page 8)
Decodes and returns the object graph previously encoded by `NSKeyedArchiver` and stored in a given `NSData` object.
- + [unarchiveObjectWithFile:](#) (page 9)
Decodes and returns the object graph previously encoded by `NSKeyedArchiver` written to the file at a given path.

Decoding Data

- [containsValueForKey:](#) (page 10)
Returns a Boolean value that indicates whether the archive contains a value for a given key within the current decoding scope.
- [decodeBoolForKey:](#) (page 11)
Decodes a Boolean value associated with a given key.
- [decodeBytesForKey:returnedLength:](#) (page 11)
Decodes a stream of bytes associated with a given key.
- [decodeDoubleForKey:](#) (page 12)
Decodes a double-precision floating-point value associated with a given key.
- [decodeFloatForKey:](#) (page 12)
Decodes a single-precision floating-point value associated with a given key.
- [decodeIntForKey:](#) (page 13)
Decodes an integer value associated with a given key.
- [decodeInt32ForKey:](#) (page 13)
Decodes a 32-bit integer value associated with a given key.
- [decodeInt64ForKey:](#) (page 13)
Decodes a 64-bit integer value associated with a given key.
- [decodeObjectForKey:](#) (page 14)
Decodes and returns an object associated with a given key.
- [finishDecoding:](#) (page 15)
Tells the receiver that you are finished decoding objects.

Managing the Delegate

- `delegate` (page 14)
Returns the receiver's delegate.
- `setDelegate:` (page 16)
Sets the receiver's delegate.

Managing Class Names

- + `setClass:forClassName:` (page 8)
Adds a class translation mapping to `NSKeyedUnarchiver` whereby objects encoded with a given class name are decoded as instances of a given class instead.
- + `classForClassName:` (page 7)
Returns the class from which `NSKeyedUnarchiver` instantiates an encoded object with a given class name.
- `setClass:forClassName:` (page 16)
Adds a class translation mapping to the receiver whereby objects encoded with a given class name are decoded as instances of a given class instead.
- `classForClassName:` (page 10)
Returns the class from which the receiver instantiates an encoded object with a given class name.

Decoding Objects

- `unarchiver:cannotDecodeObjectOfClassName:originalClasses:` (page 16) *delegate method*
Informs the delegate that the class with a given name is not available during decoding.
- `unarchiver:didDecodeObject:` (page 17) *delegate method*
Informs the delegate that a given object has been decoded.
- `unarchiver:willReplaceObject:withObject:` (page 18) *delegate method*
Informs the delegate that one object is being substituted for another.

Finishing Decoding

- `unarchiverDidFinish:` (page 18) *delegate method*
Notifies the delegate that decoding has finished.
- `unarchiverWillFinish:` (page 18) *delegate method*
Notifies the delegate that decoding is about to finish.

Class Methods

`classForClassName:`

Returns the class from which `NSKeyedUnarchiver` instantiates an encoded object with a given class name.

```
+ (Class)classForClassName:(NSString *)codedName
```

Parameters

codedName

The ostensible name of a class in an archive.

Return Value

The class from which `NSKeyedUnarchiver` instantiates an object encoded with the class name *codedName*. Returns `nil` if `NSKeyedUnarchiver` does not have a translation mapping for *codedName*.

Availability

Available in Mac OS X v10.2 and later.

See Also

+ [setClass:forClassName:](#) (page 8)

- [classForClassName:](#) (page 10)

Declared In

NSKeyedArchiver.h

setClass:forClassName:

Adds a class translation mapping to `NSKeyedUnarchiver` whereby objects encoded with a given class name are decoded as instances of a given class instead.

```
+ (void)setClass:(Class)cls forClassName:(NSString *)codedName
```

Parameters

cls

The class with which to replace instances of the class named *codedName*.

codedName

The ostensible name of a class in an archive.

Discussion

When decoding, the class's translation mapping is used only if no translation is found first in an instance's separate translation map.

Availability

Available in Mac OS X v10.2 and later.

See Also

+ [classForClassName:](#) (page 7)

- [setClass:forClassName:](#) (page 16)

Declared In

NSKeyedArchiver.h

unarchiveObjectWithData:

Decodes and returns the object graph previously encoded by `NSKeyedArchiver` and stored in a given `NSData` object.

```
+ (id)unarchiveObjectWithData:(NSData *)data
```


Parameters

data

An object graph previously encoded by `NSKeyedArchiver`.

Return Value

The object graph previously encoded by `NSKeyedArchiver` and stored in *data*.

Discussion

This method raises an `NSInvalidArchiveOperationException` if *data* is not a valid archive.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

CoreRecipes

CustomAtomicStoreSubclass

iSpend

QTQuartzPlayer

Squiggles

Declared In

`NSKeyedArchiver.h`

unarchiveObjectWithFile:

Decodes and returns the object graph previously encoded by `NSKeyedArchiver` written to the file at a given *path*.

```
+ (id)unarchiveObjectWithFile:(NSString *)path
```

Parameters

path

A path to a file that contains an object graph previously encoded by `NSKeyedArchiver`.

Return Value

The object graph previously encoded by `NSKeyedArchiver` written to the file *path*. Returns `nil` if there is no file at *path*.

Discussion

This method raises an `NSInvalidArgumentException` if the file at *path* does not contain a valid archive.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSKeyedArchiver.h`

Instance Methods

classForClassName:

Returns the class from which the receiver instantiates an encoded object with a given class name.

```
- (Class)classForClassName:(NSString *)codedName
```

Parameters

codedName

The name of a class.

Return Value

The class from which the receiver instantiates an encoded object with the class name *codedName*. Returns *nil* if the receiver does not have a translation mapping for *codedName*.

Discussion

The class's separate translation map is not searched.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [setClass:forClassName:](#) (page 16)

+ [classForClassName:](#) (page 7)

Declared In

NSKeyedArchiver.h

containsValueForKey:

Returns a Boolean value that indicates whether the archive contains a value for a given key within the current decoding scope.

```
- (BOOL)containsValueForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be *nil*.

Return Value

YES if the archive contains a value for *key* within the current decoding scope, otherwise NO.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

decodeBoolForKey:

Decodes a Boolean value associated with a given key.

```
- (BOOL)decodeBoolForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be *nil*.

Return Value

The Boolean value associated with the key *key*. Returns *NO* if *key* does not exist.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `encodeBool:forKey:` (NSKeyedArchiver)

Declared In

NSKeyedArchiver.h

decodeBytesForKey:returnedLength:

Decodes a stream of bytes associated with a given key.

```
- (const uint8_t *)decodeBytesForKey:(NSString *)key returnedLength:(NSUInteger *)lengthp
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be *nil*.

lengthp

Upon return, contains the number of bytes returned.

Return Value

The stream of bytes associated with the key *key*. Returns *NULL* if *key* does not exist.

Discussion

The returned value is a pointer to a temporary buffer owned by the receiver. The buffer goes away with the unarchiver, not the containing autorelease pool. You must copy the bytes into your own buffer if you need the data to persist beyond the life of the receiver.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `encodeBytes:length:forKey:` (NSKeyedArchiver)

Declared In

NSKeyedArchiver.h

decodeDoubleForKey:

Decodes a double-precision floating-point value associated with a given key.

```
- (double)decodeDoubleForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be *nil*.

Return Value

The double-precision floating-point value associated with the key *key*. Returns 0.0 if *key* does not exist.

Discussion

If the archived value was encoded as single-precision, the type is coerced.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `encodeDouble:forKey:` (NSKeyedArchiver)
- `encodeFloat:forKey:` (NSKeyedArchiver)

Declared In

NSKeyedArchiver.h

decodeFloatForKey:

Decodes a single-precision floating-point value associated with a given key.

```
- (float)decodeFloatForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be *nil*.

Return Value

The single-precision floating-point value associated with the key *key*. Returns 0.0 if *key* does not exist.

Discussion

If the archived value was encoded as double precision, the type is coerced, losing precision. If the archived value is too large for single precision, the method raises an `NSRangeException`.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `encodeFloat:forKey:` (NSKeyedArchiver)
- `encodeDouble:forKey:` (NSKeyedArchiver)

Declared In

NSKeyedArchiver.h

decodeInt32ForKey:

Decodes a 32-bit integer value associated with a given key.

```
- (int32_t)decodeInt32ForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be `nil`.

Return Value

The 32-bit integer value associated with the key *key*. Returns 0 if *key* does not exist.

Discussion

If the archived value was encoded with a different size but is still an integer, the type is coerced. If the archived value is too large to fit into a 32-bit integer, the method raises an `NSRangeException`.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `encodeInt32:forKey:` (NSKeyedArchiver)

Declared In

NSKeyedArchiver.h

decodeInt64ForKey:

Decodes a 64-bit integer value associated with a given key.

```
- (int64_t)decodeInt64ForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be `nil`.

Return Value

The 64-bit integer value associated with the key *key*. Returns 0 if *key* does not exist.

Discussion

If the archived value was encoded with a different size but is still an integer, the type is coerced.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `encodeInt64:forKey:` (NSKeyedArchiver)

Declared In

NSKeyedArchiver.h

decodeIntForKey:

Decodes an integer value associated with a given key.

```
- (int)decodeIntForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be `nil`.

Return Value

The integer value associated with the key *key*. Returns 0 if *key* does not exist.

Discussion

If the archived value was encoded with a different size but is still an integer, the type is coerced. If the archived value is too large to fit into the default size for an integer, the method raises an `NSRangeException`.

Availability

Available in Mac OS X v10.2 and later.

See Also

```
- encodeInt:forKey: (NSKeyedArchiver)
```

Declared In

`NSKeyedArchiver.h`

decodeObjectForKey:

Decodes and returns an object associated with a given key.

```
- (id)decodeObjectForKey:(NSString *)key
```

Parameters

key

A key in the archive within the current decoding scope. *key* must not be `nil`.

Return Value

The object associated with the key *key*. Returns `nil` if *key* does not exist, or if the value for *key* is `nil`.

Availability

Available in Mac OS X v10.2 and later.

See Also

```
- encodeObject:forKey: (NSKeyedArchiver)
```

Declared In

`NSKeyedArchiver.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value

The receiver's delegate.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [setDelegate:](#) (page 16)

Declared In

NSKeyedArchiver.h

finishDecoding

Tells the receiver that you are finished decoding objects.

- (void)finishDecoding

Discussion

Invoking this method allows the receiver to notify its delegate and to perform any final operations on the archive. Once this method is invoked, the receiver cannot decode any further values.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

initWithReadingWithData:

Initializes the receiver for decoding an archive previously encoded by `NSKeyedArchiver`.

- (id)initWithReadingWithData:(NSData *)data

Parameters

data

An archive previously encoded by `NSKeyedArchiver`.

Return Value

An `NSKeyedUnarchiver` object initialized for decoding *data*.

Discussion

When you finish decoding data, you should invoke [finishDecoding](#) (page 15).

This method raises an `NSInvalidArchiveOperationException` if *data* is not a valid archive.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

setClass:forClassName:

Adds a class translation mapping to the receiver whereby objects encoded with a given class name are decoded as instances of a given class instead.

```
- (void)setClass:(Class)c1s forClassName:(NSString *)codedName
```

Parameters

c1s

The class with which to replace instances of the class named *codedName*.

codedName

The ostensible name of a class in an archive.

Discussion

When decoding, the receiver's translation map overrides any translation that may also be present in the class's map (see [setClass:forClassName:](#) (page 8)).

Availability

Available in Mac OS X v10.2 and later.

See Also

- [classForClassName:](#) (page 10)

+ [setClass:forClassName:](#) (page 8)

Declared In

NSKeyedArchiver.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate

The delegate for the receiver.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [delegate](#) (page 14)

Declared In

NSKeyedArchiver.h

Delegate Methods

unarchiver:cannotDecodeObjectOfClassName:originalClasses:

Informs the delegate that the class with a given name is not available during decoding.


```
- (Class)unarchiver:(NSKeyedUnarchiver *)unarchiver
  cannotDecodeObjectOfClassName:(NSString *)name originalClasses:(NSArray
*)classNames
```

Parameters*unarchiver*

An unarchiver for which the receiver is the delegate.

name

The name of the class of an object *unarchiver* is trying to decode.

classNames

An array describing the class hierarchy of the encoded object, where the first element is the class name string of the encoded object, the second element is the class name of its immediate superclass, and so on.

Return Value

The class *unarchiver* should use in place of the class named *name*.

Discussion

The delegate may, for example, load some code to introduce the class to the runtime and return the class, or substitute a different class object. If the delegate returns `nil`, unarchiving aborts and the method raises an `NSInvalidUnarchiveOperationException`.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSKeyedArchiver.h`

unarchiver:didDecodeObject:

Informs the delegate that a given object has been decoded.

```
- (id)unarchiver:(NSKeyedUnarchiver *)unarchiver didDecodeObject:(id)object
```

Parameters*unarchiver*

An unarchiver for which the receiver is the delegate.

object

The object that has been decoded. *object* may be `nil`.

Return Value

The object to use in place of *object*. The delegate can either return *object* or return a different object to replace the decoded one. If the delegate returns `nil`, `nil` is the result of decoding *object*.

Discussion

This method is called after *object* has been sent `initWithCoder:` and `awakeAfterUsingCoder:`.

The delegate may use this method to keep track of the decoded objects.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSKeyedArchiver.h`

unarchiver:willReplaceObject:withObject:

Informs the delegate that one object is being substituted for another.

```
- (void)unarchiver:(NSKeyedUnarchiver *)unarchiver willReplaceObject:(id)object
withObject:(id)newObject
```

Parameters

unarchiver

An unarchiver for which the receiver is the delegate.

object

An object in the archive.

newObject

The object with which *unarchiver* will replace *object*.

Discussion

This method is called even when the delegate itself is doing, or has done, the substitution with [unarchiver:didDecodeObject:](#) (page 17).

The delegate may use this method if it is keeping track of the encoded or decoded objects.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

unarchiverDidFinish:

Notifies the delegate that decoding has finished.

```
- (void)unarchiverDidFinish:(NSKeyedUnarchiver *)unarchiver
```

Parameters

unarchiver

An unarchiver for which the receiver is the delegate.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

unarchiverWillFinish:

Notifies the delegate that decoding is about to finish.

```
- (void)unarchiverWillFinish:(NSKeyedUnarchiver *)unarchiver
```

Parameters

unarchiver

An unarchiver for which the receiver is the delegate.

Availability

Available in Mac OS X v10.2 and later.

Declared In

NSKeyedArchiver.h

Constants

Keyed Unarchiving Exception Names

Names of exceptions that are raised by `NSKeyedUnarchiver` if there is a problem extracting an archive.

```
extern NSString *NSInvalidUnarchiveOperationException;
```

Constants

`NSInvalidUnarchiveOperationException`

The name of the exception raised by `NSKeyedArchiver` if there is a problem extracting an archive.

Available in Mac OS X v10.2 and later.

Declared in `NSKeyedArchiver.h`.

Declared In

`NSKeyedUnarchiver.h`

Document Revision History

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

Date	Notes
2008-10-15	Added notes that keys must not be nil.
2007-01-18	Added definition of <code>NSInvalidUnarchiveOperationException</code> .
	If this revision does anything other than provide a Leopard version of 1.1, then the revision history for 1.1 should be included here and a new revision summary should be provided for 2.1 in Messier.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

classForClassName: **class method** [7](#)
classForClassName: **instance method** [10](#)
containsValueForKey: **instance method** [10](#)

D

decodeBoolForKey: **instance method** [11](#)
decodeBytesForKey:returnedLength: **instance method** [11](#)
decodeDoubleForKey: **instance method** [12](#)
decodeFloatForKey: **instance method** [12](#)
decodeInt32ForKey: **instance method** [13](#)
decodeInt64ForKey: **instance method** [13](#)
decodeIntForKey: **instance method** [13](#)
decodeObjectForKey: **instance method** [14](#)
delegate **instance method** [14](#)

F

finishDecoding **instance method** [15](#)

I

initWithReadingWithData: **instance method** [15](#)

K

Keyed Unarchiving Exception Names [19](#)

N

NSInvalidUnarchiveOperationException **constant** [19](#)

S

setClass:forClassName: **class method** [8](#)
setClass:forClassName: **instance method** [16](#)
setDelegate: **instance method** [16](#)

U

unarchiveObjectWithData: **class method** [8](#)
unarchiveObjectWithFile: **class method** [9](#)
unarchiver:cannotDecodeObjectOfClassName:
originalClasses: <NSObject> **delegate method** [16](#)
unarchiver:didDecodeObject: <NSObject> **delegate method** [17](#)
unarchiver:willReplaceObject:withObject:
<NSObject> **delegate method** [18](#)
unarchiverDidFinish: <NSObject> **delegate method** [18](#)
unarchiverWillFinish: <NSObject> **delegate method** [18](#)