# NSUnarchiver Class Reference

Cocoa > Data Management

2006-05-23

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

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

# NSUnarchiver Class Reference

| | |
|---|---|
| **Inherits from** | NSCoder : NSObject |
| **Conforms to** | NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/Foundation.framework |
| **Availability** | Available in Mac OS X v10.0 and later. |
| **Companion guide** | Archives and Serializations Programming Guide for Cocoa |
| **Declared in** | NSArchiver.h |
| **Related sample code** | Departments and Employees<br>MenuItemView<br>QTMetadataEditor<br>Sketch-112<br>StickiesExample |

## Overview

`NSUnarchiver`, a concrete subclass of `NSCoder`, defines methods for decoding a set of Objective-C objects from an archive. Such archives are produced by objects of the `NSArchiver` class.

In Mac OS X v10.2 and later, `NSArchiver` and `NSUnarchiver` have been replaced by `NSKeyedArchiver` and `NSKeyedUnarchiver` respectively—see *Archives and Serializations Programming Guide for Cocoa*.

## Tasks

### Initializing an NSUnarchiver

`– initForReadingWithData:` (page 10)
Returns an `NSUnarchiver` object initialized to read an archive from a given data object.

### Decoding Objects

`+ unarchiveObjectWithData:` (page 8)
Decodes and returns the object archived in a given `NSData` object.

+ `unarchiveObjectWithFile:` (page 8)
  Decodes and returns the object archived in the file *path*.

### Managing an NSUnarchiver

- `isAtEnd` (page 10)
  Returns a Boolean value that indicates whether the receiver has reached the end of the encoded data while decoding.
- `objectZone` (page 11)
  Returns the memory zone used to allocate decoded objects.
- `setObjectZone:` (page 12)
  Sets the memory zone used to allocate decoded objects.
- `systemVersion` (page 12)
  Returns the system version number in effect when the archive was created.

### Substituting Classes or Objects

+ `classNameDecodedForArchiveClassName:` (page 6)
  Returns the name of the class used when instantiating objects whose ostensible class, according to the archived data, is a given name.
+ `decodeClassName:asClassName:` (page 7)
  Instructs instances of `NSUnarchiver` to use the class with a given name when instantiating objects whose ostensible class, according to the archived data, is another given name.
- `classNameDecodedForArchiveClassName:` (page 9)
  Returns the name of the class that will be used when instantiating objects whose ostensible class, according to the archived data, is a given name.
- `decodeClassName:asClassName:` (page 9)
  Instructs the receiver to use the class with a given name when instantiating objects whose ostensible class, according to the archived data, is another given name.
- `replaceObject:withObject:` (page 11)
  Causes the receiver to substitute one given object for another whenever the latter is extracted from the archive.

## Class Methods

### classNameDecodedForArchiveClassName:

Returns the name of the class used when instantiating objects whose ostensible class, according to the archived data, is a given name.

```
+ (NSString *)classNameDecodedForArchiveClassName:(NSString *)nameInArchive
```

### Parameters

*nameInArchive*

The name of a class.

### Return Value

The name of the class used when instantiating objects whose ostensible class, according to the archived data, is *nameInArchive*. Returns *nameInArchive* if no substitute name has been specified using the class method (not the instance method) `decodeClassName:asClassName:` (page 7).

### Discussion

Note that each individual instance of `NSUnarchiver` can be given its own class name mappings by invoking the instance method `decodeClassName:asClassName:` (page 9). The `NSUnarchiver` class has no information about these instance-specific mappings, however, so they don't affect the return value of `classNameDecodedForArchiveClassName:`.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

`- classNameDecodedForArchiveClassName:` (page 9)

### Declared In

`NSArchiver.h`

## decodeClassName:asClassName:

Instructs instances of `NSUnarchiver` to use the class with a given name when instantiating objects whose ostensible class, according to the archived data, is another given name.

```
+ (void)decodeClassName:(NSString *)nameInArchive asClassName:(NSString *)trueName
```

### Parameters

*nameInArchive*

The ostensible name of a class in an archive.

*trueName*

The name of the class to use when instantiating objects whose ostensible class, according to the archived data, is *nameInArchive*.

### Discussion

This method enables easy conversion of unarchived data when the name of a class has changed since the archive was created.

Note that there is also an instance method of the same name. An instance of `NSUnarchiver` can maintain its own mapping of class names. However, if both the class method and the instance method have been invoked using an identical value for *nameInArchive*, the class method takes precedence.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

`+ classNameDecodedForArchiveClassName:` (page 6)

`- decodeClassName:asClassName:` (page 9)

**Declared In**
`NSArchiver.h`

## unarchiveObjectWithData:

Decodes and returns the object archived in a given `NSData` object.

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

**Parameters**

*data*

An `NSData` object that contains an archive created using `NSArchiver`.

**Return Value**
The object, or object graph, that was archived in *data*. Returns `nil` if *data* cannot be unarchived.

**Discussion**
This method invokes `initForReadingWithData:` (page 10) and `decodeObject` to create a temporary `NSUnarchiver` object that decodes the object. If the archived object is the root of a graph of objects, the entire graph is unarchived.

**Availability**
Available in Mac OS X v10.0 and later.

**See Also**
`encodeRootObject:` (`NSArchiver`)

**Related Sample Code**
Departments and Employees
MenuItemView
QTMetadataEditor
Sketch-112
StickiesExample

**Declared In**
`NSArchiver.h`

## unarchiveObjectWithFile:

Decodes and returns the object archived in the file *path*.

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

**Parameters**

*path*

The path to a file than contains an archive created using `NSArchiver`.

**Return Value**
The object, or object graph, that was archived in the file at *path*. Returns `nil` if the file at *path* cannot be unarchived.

**Discussion**
This convenience method reads the file by invoking the `NSData` method `dataWithContentsOfFile:` and then invokes `unarchiveObjectWithData:` (page 8).

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`NSArchiver.h`

# Instance Methods

## classNameDecodedForArchiveClassName:

Returns the name of the class that will be used when instantiating objects whose ostensible class, according to the archived data, is a given name.

```
- (NSString *)classNameDecodedForArchiveClassName:(NSString *)nameInArchive
```

**Parameters**

*nameInArchive*
The ostensible name of a class in an archive.

**Return Value**
The name of the class that will be used when instantiating objects whose ostensible class, according to the archived data, is *nameInArchive*. Returns *nameInArchive* unless a substitute name has been specified using the instance method (not the class method) `decodeClassName:asClassName:` (page 9).

**Availability**
Available in Mac OS X v10.0 and later.

**See Also**
`+ classNameDecodedForArchiveClassName:` (page 6)

**Declared In**
`NSArchiver.h`

## decodeClassName:asClassName:

Instructs the receiver to use the class with a given name when instantiating objects whose ostensible class, according to the archived data, is another given name.

```
- (void)decodeClassName:(NSString *)nameInArchive asClassName:(NSString *)trueName
```

**Parameters**

*nameInArchive*
The ostensible name of a class in an archive.

*trueName*
The name of the class to use when instantiating objects whose ostensible class, according to the archived data, is *nameInArchive*.

**Discussion**
This method enables easy conversion of unarchived data when the name of a class has changed since the archive was created.

Note that there's also a class method of the same name. The class method has precedence in case of conflicts.

**Availability**
Available in Mac OS X v10.0 and later.

**See Also**
`– classNameDecodedForArchiveClassName:` (page 9)
`+ decodeClassName:asClassName:` (page 7)

**Declared In**
`NSArchiver.h`

## initForReadingWithData:

Returns an `NSUnarchiver` object initialized to read an archive from a given data object.

```
- (id)initForReadingWithData:(NSData *)data
```

**Parameters**

*`data`*
The archive data.

**Return Value**
An `NSUnarchiver` object initialized to read an archive from *`data`*. Returns `nil` if *`data`* is not a valid archive.

**Discussion**
The method decodes the system version number that was archived in *`data`* prepares the `NSUnarchiver` object for a subsequent invocation of `decodeObject`.

Raises an `NSInvalidArgumentException` if *`data`* is `nil`.

**Availability**
Available in Mac OS X v10.0 and later.

**See Also**
`– systemVersion` (page 12)

**Declared In**
`NSArchiver.h`

## isAtEnd

Returns a Boolean value that indicates whether the receiver has reached the end of the encoded data while decoding.

```
- (BOOL)isAtEnd
```

**Return Value**
`YES` if the receiver has reached the end of the encoded data while decoding, otherwise `NO`.

**Discussion**
You can invoke this method after invoking `decodeObject` to discover whether the archive contains extra data following the encoded object graph. If it does, you can either ignore this anomaly or consider it an error.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`NSArchiver.h`

## objectZone

Returns the memory zone used to allocate decoded objects.

```
- (NSZone *)objectZone
```

**Return Value**
The memory zone used to allocate decoded objects.

**Availability**
Available in Mac OS X v10.0 and later.

**See Also**
`– setObjectZone:` (page 12)

**Declared In**
`NSArchiver.h`

## replaceObject:withObject:

Causes the receiver to substitute one given object for another whenever the latter is extracted from the archive.

```
- (void)replaceObject:(id)object withObject:(id)newObject
```

**Parameters**

*`object`*

The archived object to replace.

*`newObject`*

The object with which to replace *`object`*.

**Discussion**
*`newObject`* can be of a different class from object, and the class mappings set by `classNameDecodedForArchiveClassName:` (page 6) and `decodeClassName:asClassName:` (page 9) are ignored.

**Availability**
Available in Mac OS X v10.0 and later.

**Declared In**
`NSArchiver.h`

## setObjectZone:

Sets the memory zone used to allocate decoded objects.

```
- (void)setObjectZone:(NSZone *)zone
```

### Parameters

*zone*

The memory zone used to allocate decoded objects.

### Discussion

If *zone* is `nil`, or if this method is never invoked, the default zone is used, as given by `NSDefaultMallocZone()`.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

`- objectZone` (page 11)

### Declared In

`NSArchiver.h`

## systemVersion

Returns the system version number in effect when the archive was created.

```
- (unsigned)systemVersion
```

### Return Value

The system version number in effect when the archive was created.

### Discussion

This information is available as soon as the receiver has been initialized.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`NSArchiver.h`

# Document Revision History

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

| Date | Notes |
|---|---|
| 2006-05-23 | First publication of this content as a separate document. |

# Index