

---

# NSCoder Class Reference

[Cocoa > Data Management](#)



2006-07-23



Apple Inc.  
© 2006 Apple Computer, Inc.  
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

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

iPhone is a trademark of Apple Inc.

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

## NSCoder Class Reference 5

---

Overview	5
Tasks	6
Testing Coder	6
Encoding Data	6
Decoding Data	7
Managing Zones	9
Getting Version Information	9
Instance Methods	9
allowsKeyedCoding	9
containsValueForKey:	10
decodeArrayOfObjCType:count:at:	10
decodeBoolForKey:	10
decodeBytesForKey:returnedLength:	11
decodeBytesWithReturnedLength:	11
decodeDataObject	12
decodeDoubleForKey:	12
decodeFloatForKey:	12
decodeInt32ForKey:	13
decodeInt64ForKey:	13
decodeIntegerForKey:	14
decodeIntForKey:	14
decodeObject	14
decodeObjectForKey:	15
decodePoint	15
decodePointForKey:	16
decodePropertyList	16
decodeRect	16
decodeRectForKey:	16
decodeSize	17
decodeSizeForKey:	17
decodeValueOfObjCType:at:	17
decodeValuesOfObjCTypes:	18
encodeArrayOfObjCType:count:at:	18
encodeBool:forKey:	19
encodeBycopyObject:	19
encodeByrefObject:	20
encodeBytes:length:	20
encodeBytes:length:forKey:	21
encodeConditionalObject:	21
encodeConditionalObject:forKey:	21

- encodeDataObject: 22
- encodeDouble:forKey: 22
- encodeFloat:forKey: 23
- encodeInt32:forKey: 23
- encodeInt64:forKey: 24
- encodeInt:forKey: 24
- encodeInteger:forKey: 24
- encodeObject: 25
- encodeObject:forKey: 25
- encodePoint: 26
- encodePoint:forKey: 26
- encodePropertyList: 27
- encodeRect: 27
- encodeRect:forKey: 27
- encodeRootObject: 27
- encodeSize: 28
- encodeSize:forKey: 28
- encodeValueOfObjCType:at: 29
- encodeValuesOfObjCTypes: 29
- objectZone 30
- setObjectZone: 30
- systemVersion 30
- versionForClassName: 31

**Appendix A      Deprecated NSCoder Methods 33**

---

- Deprecated in Mac OS X v10.5 33
- decodeNXObject 33
- encodeNXObject: 33

**Document Revision History 35**

---

**Index 37**

---

# NSCoder Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Archives and Serializations Programming Guide for Cocoa
<b>Declared in</b>	NSCoder.h NSGeometry.h NSKeyedArchiver.h
<b>Related sample code</b>	IBFragmentView iSpend Mountains Reducer StickiesExample

## Overview

The `NSCoder` abstract class declares the interface used by concrete subclasses to transfer objects and other Objective-C data items between memory and some other format. This capability provides the basis for archiving (where objects and data items are stored on disk) and distribution (where objects and data items are copied between different processes or threads). The concrete subclasses provided by Foundation for these purposes are `NSArchiver`, `NSUnarchiver`, `NSKeyedArchiver`, `NSKeyedUnarchiver`, and `NSPortCoder`. Concrete subclasses of `NSCoder` are referred to in general as coder classes, and instances of these classes as coder objects (or simply coders). A coder object that can only encode values is referred to as an encoder object, and one that can only decode values as a decoder object.

`NSCoder` operates on objects, scalars, C arrays, structures, and strings, and on pointers to these types. It does not handle types whose implementation varies across platforms, such as `union`, `void *`, function pointers, and long chains of pointers. A coder object stores object type information along with the data, so an object decoded from a stream of bytes is normally of the same class as the object that was originally encoded into the stream. An object can change its class when encoded, however; this is described in *Archives and Serializations Programming Guide for Cocoa*.

## Tasks

### Testing Coder

- [allowsKeyedCoding](#) (page 9)  
Returns a Boolean value that indicates whether the receiver supports keyed coding of objects.
- [containsValueForKey:](#) (page 10)  
Returns a Boolean value that indicates whether an encoded value is available for a string.

### Encoding Data

- [encodeArrayOfObjCType:count:at:](#) (page 18)  
Encodes an array of *count* items, whose Objective-C type is given by *itemType*.
- [encodeBool:forKey:](#) (page 19)  
Encodes *boolv* and associates it with the string *key*.
- [encodeBycopyObject:](#) (page 19)  
Can be overridden by subclasses to encode *object* so that a copy, rather than a proxy, is created upon decoding.
- [encodeByrefObject:](#) (page 20)  
Can be overridden by subclasses to encode *object* so that a proxy, rather than a copy, is created upon decoding.
- [encodeBytes:length:](#) (page 20)  
Encodes a buffer of data whose types are unspecified.
- [encodeBytes:length:forKey:](#) (page 21)  
Encodes a buffer of data, *bytesp*, whose length is specified by *lenv*, and associates it with the string *key*.
- [encodeConditionalObject:](#) (page 21)  
Can be overridden by subclasses to conditionally encode *object*, preserving common references to that object.
- [encodeConditionalObject:forKey:](#) (page 21)  
Conditionally encodes a reference to *objv* and associates it with the string *key* only if *objv* has been unconditionally encoded with [encodeObject:forKey:](#) (page 25).
- [encodeDataObject:](#) (page 22)  
Encodes a given NSData object.
- [encodeDouble:forKey:](#) (page 22)  
Encodes *realv* and associates it with the string *key*.
- [encodeFloat:forKey:](#) (page 23)  
Encodes *realv* and associates it with the string *key*.
- [encodeInt:forKey:](#) (page 24)  
Encodes *intv* and associates it with the string *key*.
- [encodeInteger:forKey:](#) (page 24)  
Encodes a given NSInteger and associates it with a given key.

- [encodeInt32:forKey:](#) (page 23)  
Encodes the 32-bit integer *intv* and associates it with the string *key*.
- [encodeInt64:forKey:](#) (page 24)  
Encodes the 64-bit integer *intv* and associates it with the string *key*.
- [encodeObject:](#) (page 25)  
Encodes *object*.
- [encodeObject:forKey:](#) (page 25)  
Encodes the object *objv* and associates it with the string *key*.
- [encodePoint:](#) (page 26)  
Encodes *point*.
- [encodePoint:forKey:](#) (page 26)  
Encodes *point* and associates it with the string *key*.
- [encodePropertyList:](#) (page 27)  
Encodes the property list *aPropertyList*.
- [encodeRect:](#) (page 27)  
Encodes *rect*.
- [encodeRect:forKey:](#) (page 27)  
Encodes *rect* and associates it with the string *key*.
- [encodeRootObject:](#) (page 27)  
Can be overridden by subclasses to encode an interconnected group of Objective-C objects, starting with *rootObject*.
- [encodeSize:](#) (page 28)  
Encodes *size*.
- [encodeSize:forKey:](#) (page 28)  
Encodes *size* and associates it with the string *key*.
- [encodeValueOfObjCType:at:](#) (page 29)  
Must be overridden by subclasses to encode a single value residing at *address*, whose Objective-C type is given by *valueType*.
- [encodeValuesOfObjCTypes:](#) (page 29)  
Encodes a series of values of potentially differing Objective-C types.
- [encodeNXObject:](#) (page 33) **Deprecated in Mac OS X v10.5**  
Encodes an old-style object onto the coder.

## Decoding Data

- [decodeArrayOfObjCType:count:at:](#) (page 10)  
Decodes an array of *count* items, whose Objective-C type is given by *itemType*.
- [decodeBoolForKey:](#) (page 10)  
Decodes and returns a boolean value that was previously encoded with [encodeBool:forKey:](#) (page 19) and associated with the string *key*.
- [decodeBytesForKey:returnedLength:](#) (page 11)  
Decodes a buffer of data that was previously encoded with [encodeBytes:length:forKey:](#) (page 21) and associated with the string *key*.

- [decodeBytesWithReturnedLength:](#) (page 11)  
Decodes a buffer of data whose types are unspecified.
- [decodeDataObject](#) (page 12)  
Decodes and returns an NSData object that was previously encoded with [encodeDataObject:](#) (page 22). Subclasses must override this method.
- [decodeDoubleForKey:](#) (page 12)  
Decodes and returns a double value that was previously encoded with either [encodeFloat:forKey:](#) (page 23) or [encodeDouble:forKey:](#) (page 22) and associated with the string *key*.
- [decodeFloatForKey:](#) (page 12)  
Decodes and returns a float value that was previously encoded with [encodeFloat:forKey:](#) (page 23) or [encodeDouble:forKey:](#) (page 22) and associated with the string *key*.
- [decodeIntForKey:](#) (page 14)  
Decodes and returns an int value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string *key*.
- [decodeIntegerForKey:](#) (page 14)  
Decodes and returns an NSInteger value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string *key*.
- [decodeInt32ForKey:](#) (page 13)  
Decodes and returns a 32-bit integer value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string *key*.
- [decodeInt64ForKey:](#) (page 13)  
Decodes and returns a 64-bit integer value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string *key*.
- [decodeObject](#) (page 14)  
Decodes an Objective-C object that was previously encoded with any of the `encode...Object:` methods.
- [decodeObjectForKey:](#) (page 15)  
Decodes and returns an autoreleased Objective-C object that was previously encoded with [encodeObject:forKey:](#) (page 25) or [encodeConditionalObject:forKey:](#) (page 21) and associated with the string *key*.
- [decodePoint](#) (page 15)  
Decodes and returns an NSPoint structure that was previously encoded with [encodePoint:](#) (page 26).
- [decodePointForKey:](#) (page 16)  
Decodes and returns an NSPoint structure that was previously encoded with [encodePoint:forKey:](#) (page 26).
- [decodePropertyList](#) (page 16)  
Decodes a property list that was previously encoded with [encodePropertyList:](#) (page 27).
- [decodeRect](#) (page 16)  
Decodes and returns an NSRect structure that was previously encoded with [encodeRect:](#) (page 27).

- [decodeRectForKey:](#) (page 16)  
Decodes and returns an `NSRect` structure that was previously encoded with [encodeRect:forKey:](#) (page 27).
- [decodeSize](#) (page 17)  
Decodes and returns an `NSSize` structure that was previously encoded with [encodeSize:](#) (page 28).
- [decodeSizeForKey:](#) (page 17)  
Decodes and returns an `NSSize` structure that was previously encoded with [encodeSize:forKey:](#) (page 28).
- [decodeValueOfObjCType:at:](#) (page 17)  
Decodes a single value, whose Objective-C type is given by *valueType*.
- [decodeValuesOfObjCTypes:](#) (page 18)  
Decodes a series of potentially different Objective-C types.
- [decodeNXObject](#) (page 33) **Deprecated in Mac OS X v10.5**  
Decodes an object previously written with [encodeNXObject:](#) (page 33).

## Managing Zones

- [objectZone](#) (page 30)  
Returns the memory zone used to allocate decoded objects.
- [setObjectZone:](#) (page 30)  
NSCoder's implementation of this method does nothing.

## Getting Version Information

- [systemVersion](#) (page 30)  
During encoding, this method should return the system version currently in effect.
- [versionForClassName:](#) (page 31)  
Returns the version in effect for the class with a given name.

## Instance Methods

### allowsKeyedCoding

Returns a Boolean value that indicates whether the receiver supports keyed coding of objects.

- (BOOL)allowsKeyedCoding

#### Discussion

The default implementation returns NO. Concrete subclasses that support keyed coding, such as `NSKeyedArchiver`, need to override this method to return YES.

#### Availability

Available in Mac OS X v10.2 and later.

**Declared In**

NSCoder.h

**containsValueForKey:**

Returns a Boolean value that indicates whether an encoded value is available for a string.

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

**Discussion**

The string is passed as *key*. Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSCoder.h

**decodeArrayOfObjCType:count:at:**

Decodes an array of *count* items, whose Objective-C type is given by *itemType*.

```
- (void)decodeArrayOfObjCType:(const char *)itemType count:(NSUInteger)count at:(void *)address
```

**Discussion**

The items are decoded into the buffer beginning at *address*, which must be large enough to contain them all. *itemType* must contain exactly one type code. NSCoder's implementation invokes [decodeValueOfObjCType:at:](#) (page 17) to decode the entire array of items. If you use this method to decode an array of Objective-C objects, you are responsible for releasing each object.

This method matches an [encodeArrayOfObjCType:count:at:](#) (page 18) message used during encoding.

For information on creating an Objective-C type code suitable for *itemType*, see the "Type Encodings" section in the "The Objective-C Runtime System" chapter of *The Objective-C 2.0 Programming Language*.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [decodeValuesOfObjCTypes:](#) (page 18)

**Declared In**

NSCoder.h

**decodeBoolForKey:**

Decodes and returns a boolean value that was previously encoded with [encodeBool:forKey:](#) (page 19) and associated with the string *key*.

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

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

iSpend

Reducer

**Declared In**

NSCoder.h

**decodeBytesForKey:returnedLength:**

Decodes a buffer of data that was previously encoded with [encodeBytes:length:forKey:](#) (page 21) and associated with the string *key*.

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

**Discussion**

The buffer's length is returned by reference in *lengthp*. The returned bytes are immutable. Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [encodeBytes:length:forKey:](#) (page 21)

**Declared In**

NSCoder.h

**decodeBytesWithReturnedLength:**

Decodes a buffer of data whose types are unspecified.

```
- (void *)decodeBytesWithReturnedLength:(NSUInteger *)numBytes
```

**Discussion**

NSCoder's implementation invokes [decodeValueOfObjCType:at:](#) (page 17) to decode the data as a series of bytes, which this method then places into a buffer and returns. The buffer's length is returned by reference in *numBytes*. If you need the bytes beyond the scope of the current autorelease pool, you must copy them.

This method matches an [encodeBytes:length:](#) (page 20) message used during encoding.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [encodeArrayOfObjCType:count:at:](#) (page 18)

**Declared In**

NSCoder.h

**decodeDataObject**

Decodes and returns an NSData object that was previously encoded with [encodeDataObject:](#) (page 22). Subclasses must override this method.

```
- (NSData *)decodeDataObject
```

**Discussion**

The implementation of your overriding method must match the implementation of your [encodeDataObject:](#) (page 22) method. For example, a typical [encodeDataObject:](#) (page 22) method encodes the number of bytes of data followed by the bytes themselves. Your override of this method must read the number of bytes, create an NSData object of the appropriate size, and decode the bytes into the new NSData object. Your overriding method should return an autoreleased NSData object.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSCoder.h

**decodeDoubleForKey:**

Decodes and returns a double value that was previously encoded with either [encodeFloat:forKey:](#) (page 23) or [encodeDouble:forKey:](#) (page 22) and associated with the string *key*.

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

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

QTQuartzPlayer

Squiggles

**Declared In**

NSCoder.h

**decodeFloatForKey:**

Decodes and returns a float value that was previously encoded with [encodeFloat:forKey:](#) (page 23) or [encodeDouble:forKey:](#) (page 22) and associated with the string *key*.

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

**Discussion**

If the value was encoded as a `double`, the extra precision is lost. Also, if the encoded real value does not fit into a `float`, the method raises an `NSRangeException`. Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

iSpend

**Declared In**

NSCoder.h

**decodeInt32ForKey:**

Decodes and returns a 32-bit integer value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string *key*.

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

**Discussion**

If the encoded integer does not fit into a 32-bit integer, the method raises an `NSRangeException`. Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSCoder.h

**decodeInt64ForKey:**

Decodes and returns a 64-bit integer value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string *key*.

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

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSCoder.h

## decodeIntegerForKey:

Decodes and returns an `NSInteger` value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string `key`.

```
- (NSInteger)decodeIntegerForKey:(NSString *)key
```

### Discussion

If the encoded integer does not fit into the `NSInteger` size, the method raises an `NSRangeException`. Subclasses must override this method if they perform keyed coding.

### Availability

Available in Mac OS X v10.5 and later.

### Declared In

`NSCoder.h`

## decodeIntForKey:

Decodes and returns an `int` value that was previously encoded with [encodeInt:forKey:](#) (page 24), [encodeInteger:forKey:](#) (page 24), [encodeInt32:forKey:](#) (page 23), or [encodeInt64:forKey:](#) (page 24) and associated with the string `key`.

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

### Discussion

If the encoded integer does not fit into the default integer size, the method raises an `NSRangeException`. Subclasses must override this method if they perform keyed coding.

### Availability

Available in Mac OS X v10.2 and later.

### Related Sample Code

Reducer

### Declared In

`NSCoder.h`

## decodeObject

Decodes an Objective-C object that was previously encoded with any of the `encode...Object:` methods.

```
- (id)decodeObject
```

### Discussion

`NSCoder`'s implementation invokes [decodeValueOfObjCType:at:](#) (page 17) to decode the object data.

Subclasses may need to override this method if they override any of the corresponding `encode...Object:` methods. For example, if an object was encoded conditionally using the [encodeConditionalObject:](#) (page 21) method, this method needs to check whether the object had actually been encoded.

The implementation for the concrete subclass `NSUnarchiver` returns an object that is retained by the unarchiver and is released when the unarchiver is deallocated. Therefore, you must retain the returned object before releasing the unarchiver. `NSKeyedUnarchiver`'s implementation, however, returns an autoreleased object, so its life is the same as the current autorelease pool instead of the keyed unarchiver.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [encodeBycopyObject:](#) (page 19)
- [encodeByrefObject:](#) (page 20)
- [encodeObject:](#) (page 25)

**Related Sample Code**

bMoviePalette

bMoviePaletteCocoa

Clock Control

StickiesExample

**Declared In**

NSCoder.h

**decodeObjectForKey:**

Decodes and returns an autoreleased Objective-C object that was previously encoded with [encodeObject:forKey:](#) (page 25) or [encodeConditionalObject:forKey:](#) (page 21) and associated with the string *key*.

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

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

IBFragmentView

iSpend

Mountains

Squiggles

StickiesExample

**Declared In**

NSCoder.h

**decodePoint**

Decodes and returns an `NSPoint` structure that was previously encoded with [encodePoint:](#) (page 26).

```
- (NSPoint)decodePoint
```

#### **Availability**

Available in Mac OS X v10.0 and later.

#### **Declared In**

NSGeometry.h

### **decodePointForKey:**

Decodes and returns an `NSPoint` structure that was previously encoded with [encodePoint:forKey:](#) (page 26).

```
- (NSPoint)decodePointForKey:(NSString *)key
```

#### **Availability**

Available in Mac OS X v10.2 and later.

#### **Declared In**

NSKeyedArchiver.h

### **decodePropertyList**

Decodes a property list that was previously encoded with [encodePropertyList:](#) (page 27).

```
- (id)decodePropertyList
```

#### **Availability**

Available in Mac OS X v10.0 and later.

#### **Declared In**

NSCoder.h

### **decodeRect**

Decodes and returns an `NSRect` structure that was previously encoded with [encodeRect:](#) (page 27).

```
- (NSRect)decodeRect
```

#### **Availability**

Available in Mac OS X v10.0 and later.

#### **Declared In**

NSGeometry.h

### **decodeRectForKey:**

Decodes and returns an `NSRect` structure that was previously encoded with [encodeRect:forKey:](#) (page 27).

```
- (NSRect)decodeRectForKey:(NSString *)key
```

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSKeyedArchiver.h

**decodeSize**

Decodes and returns an `NSSize` structure that was previously encoded with `encodeSize:` (page 28).

```
- (NSSize)decodeSize
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSGeometry.h

**decodeSizeForKey:**

Decodes and returns an `NSSize` structure that was previously encoded with `encodeSize:forKey:` (page 28).

```
- (NSSize)decodeSizeForKey:(NSString *)key
```

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

Reducer

**Declared In**

NSKeyedArchiver.h

**decodeValueOfObjCType:at:**

Decodes a single value, whose Objective-C type is given by `valueType`.

```
- (void)decodeValueOfObjCType:(const char *)valueType at:(void *)data
```

**Discussion**

`valueType` must contain exactly one type code, and the buffer specified by `data` must be large enough to hold the value corresponding to that type code. For information on creating an Objective-C type code suitable for `valueType`, see the “Type Encodings” section in “The Objective-C Runtime System” chapter of *The Objective-C 2.0 Programming Language*.

Subclasses must override this method and provide an implementation to decode the value. In your overriding implementation, decode the value into the buffer beginning at `data`. If your overriding method is capable of decoding an Objective-C object, your method must also retain that object. Clients of this method are then responsible for releasing the object.

This method matches an `encodeValueOfObjCType:at:` (page 29) message used during encoding.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [decodeArrayOfObjCType:count:at:](#) (page 10)
- [decodeValuesOfObjCTypes:](#) (page 18)
- [decodeObject](#) (page 14)

**Declared In**

NSCoder.h

**decodeValuesOfObjCTypes:**

Decodes a series of potentially different Objective-C types.

```
- (void)decodeValuesOfObjCTypes:(const char *)valueTypes, ...
```

**Discussion**

*valueTypes* is a single string containing any number of type codes. The variable arguments to this method consist of one or more pointer arguments, each of which specifies the buffer in which to place a single decoded value. For each type code in *valueTypes*, you must specify a corresponding pointer argument whose buffer is large enough to hold the decoded value. If you use this method to decode Objective-C objects, you are responsible for releasing them.

This method matches an [encodeValuesOfObjCTypes:](#) (page 29) message used during encoding.

NSCoder's implementation invokes [decodeValueOfObjCType:at:](#) (page 17) to decode individual types. Subclasses that implement the [decodeValueOfObjCType:at:](#) (page 17) method do not need to override this method.

For information on creating Objective-C type codes suitable for *valueTypes*, see the "Type Encodings" section in "The Objective-C Runtime System" chapter of *The Objective-C 2.0 Programming Language*.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [decodeArrayOfObjCType:count:at:](#) (page 10)

**Declared In**

NSCoder.h

**encodeArrayOfObjCType:count:at:**

Encodes an array of *count* items, whose Objective-C type is given by *itemType*.

```
- (void)encodeArrayOfObjCType:(const char *)itemType count:(NSUInteger)count
  at:(const void *)address
```

**Discussion**

The values are encoded from the buffer beginning at *address*. *itemType* must contain exactly one type code. NSCoder's implementation invokes [encodeValueOfObjCType:at:](#) (page 29) to encode the entire array of items. Subclasses that implement the [encodeValueOfObjCType:at:](#) (page 29) method do not need to override this method.

This method must be matched by a subsequent [decodeArrayOfObjCType:count:at:](#) (page 10) message.

For information on creating an Objective-C type code suitable for *itemType*, see the "Type Encodings" section in "The Objective-C Runtime System" chapter of *The Objective-C Programming Language*.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [encodeValueOfObjCType:at:](#) (page 29)
- [encodeValuesOfObjCTypes:](#) (page 29)
- [encodeBytes:length:](#) (page 20)

**Declared In**

NSCoder.h

**encodeBool:forKey:**

Encodes *boolv* and associates it with the string *key*.

```
- (void)encodeBool:(BOOL)boolv forKey:(NSString *)key
```

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

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

**Related Sample Code**

iSpend  
Reducer

**Declared In**

NSCoder.h

**encodeBycopyObject:**

Can be overridden by subclasses to encode *object* so that a copy, rather than a proxy, is created upon decoding.

```
- (void)encodeBycopyObject:(id)object
```

**Discussion**

NSCoder's implementation simply invokes [encodeObject:](#) (page 25).

This method must be matched by a corresponding [decodeObject](#) (page 14) message.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [encodeRootObject](#): (page 27)
- [encodeConditionalObject](#): (page 21)
- [encodeByrefObject](#): (page 20)

#### Declared In

NSCoder.h

## encodeByrefObject:

Can be overridden by subclasses to encode *object* so that a proxy, rather than a copy, is created upon decoding.

```
- (void)encodeByrefObject:(id)object
```

#### Discussion

NSCoder's implementation simply invokes [encodeObject](#): (page 25).

This method must be matched by a corresponding [decodeObject](#) (page 14) message.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [encodeBycopyObject](#): (page 19)

#### Declared In

NSCoder.h

## encodeBytes:length:

Encodes a buffer of data whose types are unspecified.

```
- (void)encodeBytes:(const void *)address length:(NSUInteger)numBytes
```

#### Discussion

The buffer to be encoded begins at *address*, and its length in bytes is given by *numBytes*.

This method must be matched by a corresponding [decodeBytesWithReturnedLength](#): (page 11) message.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [encodeArrayOfObjCType:count:at:](#) (page 18)

#### Declared In

NSCoder.h

## encodeBytes:length:forKey:

Encodes a buffer of data, *bytesp*, whose length is specified by *length*, and associates it with the string *key*.

```
- (void)encodeBytes:(const uint8_t *)bytesp length:(NSUInteger)length forKey:(NSString *)key
```

### Discussion

Subclasses must override this method if they perform keyed coding.

### Availability

Available in Mac OS X v10.2 and later.

### See Also

- [decodeBytesForKey:returnedLength:](#) (page 11)

### Declared In

NSCoder.h

## encodeConditionalObject:

Can be overridden by subclasses to conditionally encode *object*, preserving common references to that object.

```
- (void)encodeConditionalObject:(id)object
```

### Discussion

In the overriding method, *object* should be encoded only if it's unconditionally encoded elsewhere (with any other `encode...Object:` method).

This method must be matched by a subsequent [decodeObject](#) (page 14) message. Upon decoding, if *object* was never encoded unconditionally, `decodeObject` returns `nil` in place of *object*. However, if *object* was encoded unconditionally, all references to *object* must be resolved.

NSCoder's implementation simply invokes [encodeObject:](#) (page 25).

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [encodeRootObject:](#) (page 27)
- [encodeObject:](#) (page 25)
- [encodeBycopyObject:](#) (page 19)
- `encodeConditionalObject:` (NSArchiver)

### Declared In

NSCoder.h

## encodeConditionalObject:forKey:

Conditionally encodes a reference to *objv* and associates it with the string *key* only if *objv* has been unconditionally encoded with [encodeObject:forKey:](#) (page 25).

- (void)encodeConditionalObject:(id)objv forKey:(NSString \*)key

#### Discussion

Subclasses must override this method if they support keyed coding.

The encoded object is decoded with the [decodeObjectForKey:](#) (page 15) method. If *objv* was never encoded unconditionally, [decodeObjectForKey:](#) (page 15) returns `nil` in place of *objv*.

#### Availability

Available in Mac OS X v10.2 and later.

#### Related Sample Code

IBFragmentView

Reducer

#### Declared In

NSCoder.h

## encodeDataObject:

Encodes a given NSData object.

- (void)encodeDataObject:(NSData \*)data

#### Discussion

Subclasses must override this method.

This method must be matched by a subsequent [decodeDataObject:](#) (page 12) message.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [encodeObject:](#) (page 25)

#### Declared In

NSCoder.h

## encodeDouble:forKey:

Encodes *realv* and associates it with the string *key*.

- (void)encodeDouble:(double)realv forKey:(NSString \*)key

#### Discussion

Subclasses must override this method if they perform keyed coding.

#### Availability

Available in Mac OS X v10.2 and later.

#### See Also

- [decodeDoubleForKey:](#) (page 12)

- [decodeFloatForKey:](#) (page 12)

### Related Sample Code

QTQuartzPlayer

Squiggles

### Declared In

NSCoder.h

## encodeFloat:forKey:

Encodes *realv* and associates it with the string *key*.

```
- (void)encodeFloat:(float)realv forKey:(NSString *)key
```

### Discussion

Subclasses must override this method if they perform keyed coding.

### Availability

Available in Mac OS X v10.2 and later.

### See Also

- [decodeFloatForKey:](#) (page 12)

- [decodeDoubleForKey:](#) (page 12)

### Related Sample Code

iSpend

### Declared In

NSCoder.h

## encodeInt32:forKey:

Encodes the 32-bit integer *intv* and associates it with the string *key*.

```
- (void)encodeInt32:(int32_t)intv forKey:(NSString *)key
```

### Discussion

Subclasses must override this method if they perform keyed coding.

### Availability

Available in Mac OS X v10.2 and later.

### See Also

- [decodeIntForKey:](#) (page 14)

- [decodeIntegerForKey:](#) (page 14)

- [decodeInt32ForKey:](#) (page 13)

- [decodeInt64ForKey:](#) (page 13)

### Declared In

NSCoder.h

**encodeInt64:forKey:**

Encodes the 64-bit integer *intv* and associates it with the string *key*.

```
- (void)encodeInt64:(int64_t)intv forKey:(NSString *)key
```

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [decodeIntForKey:](#) (page 14)
- [decodeIntegerForKey:](#) (page 14)
- [decodeInt32ForKey:](#) (page 13)
- [decodeInt64ForKey:](#) (page 13)

**Declared In**

NSCoder.h

**encodeInt:forKey:**

Encodes *intv* and associates it with the string *key*.

```
- (void)encodeInt:(int)intv forKey:(NSString *)key
```

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [decodeIntForKey:](#) (page 14)
- [decodeIntegerForKey:](#) (page 14)
- [decodeInt32ForKey:](#) (page 13)
- [decodeInt64ForKey:](#) (page 13)

**Related Sample Code**

Reducer

**Declared In**

NSCoder.h

**encodeInteger:forKey:**

Encodes a given `NSInteger` and associates it with a given key.

```
- (void)encodeInteger:(NSInteger)intv forKey:(NSString *)key
```

**Discussion**

Subclasses must override this method if they perform keyed coding.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- [decodeIntForKey:](#) (page 14)
- [decodeIntegerForKey:](#) (page 14)
- [decodeInt32ForKey:](#) (page 13)
- [decodeInt64ForKey:](#) (page 13)

**Declared In**

NSCoder.h

**encodeObject:**

Encodes *object*.

```
- (void)encodeObject:(id)object
```

**Discussion**

NSCoder's implementation simply invokes [encodeValueOfObjCType:at:](#) (page 29) to encode *object*. Subclasses can override this method to encode a reference to *object* instead of *object* itself. For example, `NSArchiver` detects duplicate objects and encodes a reference to the original object rather than encode the same object twice.

This method must be matched by a subsequent [decodeObject:](#) (page 14) message.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [encodeRootObject:](#) (page 27)
- [encodeConditionalObject:](#) (page 21)
- [encodeBycopyObject:](#) (page 19)

**Related Sample Code**

bMoviePalette

bMoviePaletteCocoa

Clock Control

StickiesExample

**Declared In**

NSCoder.h

**encodeObject:forKey:**

Encodes the object *objv* and associates it with the string *key*.

```
- (void)encodeObject:(id)objv forKey:(NSString *)key
```

**Discussion**

Subclasses must override this method to identify multiple encodings of *objv* and encode a reference to *objv* instead. For example, `NSKeyedArchiver` detects duplicate objects and encodes a reference to the original object rather than encode the same object twice.

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

- [decodeObjectForKey:](#) (page 15)

**Related Sample Code**

IBFragmentsView

iSpend

Mountains

Squiggles

StickiesExample

**Declared In**

`NSCoder.h`

**encodePoint:**

Encodes *point*.

```
- (void)encodePoint:(NSPoint)point
```

**Discussion**

`NSCoder`'s implementation invokes [encodeValueOfObjCType:at:](#) (page 29) to encode *point*.

This method must be matched by a subsequent [decodePoint](#) (page 15) message.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSGeometry.h`

**encodePoint:forKey:**

Encodes *point* and associates it with the string *key*.

```
- (void)encodePoint:(NSPoint)point forKey:(NSString *)key
```

**Availability**

Available in Mac OS X v10.2 and later.

**See Also**

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

**Declared In**

`NSKeyedArchiver.h`

## encodePropertyList:

Encodes the property list *aPropertyList*.

- (void)encodePropertyList:(id)aPropertyList

### Discussion

NSCoder's implementation invokes [encodeValueOfObjCType:at:](#) (page 29) to encode *aPropertyList*.

This method must be matched by a subsequent [decodePropertyList](#) (page 16) message.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

NSCoder.h

## encodeRect:

Encodes *rect*.

- (void)encodeRect:(NSRect)rect

### Discussion

NSCoder's implementation invokes [encodeValueOfObjCType:at:](#) (page 29) to encode *rect*.

This method must be matched by a subsequent [decodeRect](#) (page 16) message.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

NSGeometry.h

## encodeRect:forKey:

Encodes *rect* and associates it with the string *key*.

- (void)encodeRect:(NSRect)rect forKey:(NSString \*)key

### Availability

Available in Mac OS X v10.2 and later.

### See Also

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

### Declared In

NSKeyedArchiver.h

## encodeRootObject:

Can be overridden by subclasses to encode an interconnected group of Objective-C objects, starting with *rootObject*.

- (void)encodeRootObject:(id)rootObject

#### Discussion

NSCoder's implementation simply invokes [encodeObject:](#) (page 25).

This method must be matched by a subsequent [decodeObject:](#) (page 14) message.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [encodeObject:](#) (page 25)
- [encodeConditionalObject:](#) (page 21)
- [encodeBycopyObject:](#) (page 19)
- [encodeRootObject:](#) (NSArchiver)

#### Declared In

NSCoder.h

## encodeSize:

Encodes *size*.

- (void)encodeSize:(NSSize)size

#### Discussion

NSCoder's implementation invokes [encodeValueOfObjCType:at:](#) (page 29) to encode *size*.

This method must be matched by a subsequent [decodeSize:](#) (page 17) message.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

NSGeometry.h

## encodeSize:forKey:

Encodes *size* and associates it with the string *key*.

- (void)encodeSize:(NSSize)size forKey:(NSString \*)key

#### Availability

Available in Mac OS X v10.2 and later.

#### See Also

- [decodeSizeForKey:](#) (page 17)

#### Related Sample Code

Reducer

#### Declared In

NSKeyedArchiver.h

## encodeValueOfObjCType:at:

Must be overridden by subclasses to encode a single value residing at *address*, whose Objective-C type is given by *valueType*.

```
- (void)encodeValueOfObjCType:(const char *)valueType at:(const void *)address
```

### Discussion

*valueType* must contain exactly one type code.

This method must be matched by a subsequent [decodeValueOfObjCType:at:](#) (page 17) message.

For information on creating an Objective-C type code suitable for *valueType*, see the “Type Encodings” section in “The Objective-C Runtime System” chapter of *The Objective-C 2.0 Programming Language*.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [encodeArrayOfObjCType:count:at:](#) (page 18)
- [encodeValuesOfObjCTypes:](#) (page 29)

### Declared In

NSCoder.h

## encodeValuesOfObjCTypes:

Encodes a series of values of potentially differing Objective-C types.

```
- (void)encodeValuesOfObjCTypes:(const char *)valueTypes, ...
```

### Discussion

*valueTypes* is a single string containing any number of type codes. The variable arguments to this method consist of one or more pointer arguments, each of which specifies a buffer containing the value to be encoded. For each type code in *valueTypes*, you must specify a corresponding pointer argument.

This method must be matched by a subsequent [decodeValuesOfObjCTypes:](#) (page 18) message.

NSCoder’s implementation invokes [encodeValueOfObjCType:at:](#) (page 29) to encode individual types. Subclasses that implement the [encodeValueOfObjCType:at:](#) (page 29) method do not need to override this method. However, subclasses that provide a more efficient approach for encoding a series of values may override this method to implement that approach.

For information on creating Objective-C type codes suitable for *valueTypes*, see the “Type Encodings” section in “The Objective-C Runtime System” chapter of *The Objective-C 2.0 Programming Language*.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [encodeArrayOfObjCType:count:at:](#) (page 18)
- [encodeValueOfObjCType:at:](#) (page 29)

### Declared In

NSCoder.h

## objectZone

Returns the memory zone used to allocate decoded objects.

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

### Discussion

NSCoder's implementation simply returns the default memory zone, as given by `NSDefaultMallocZone()`.

Subclasses must override this method and the [setObjectZone:](#) (page 30) method to allow objects to be decoded into a zone other than the default zone. In its overriding implementation of this method, your subclass should return the current memory zone (if one has been set) or the default zone (if no other zone has been set).

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

NSCoder.h

## setObjectZone:

NSCoder's implementation of this method does nothing.

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

### Discussion

Can be overridden by subclasses to set the memory zone used to allocate decoded objects.

Subclasses must override this method and [objectZone](#) (page 30) to allow objects to be decoded into a zone other than the default zone. In its overriding implementation of this method, your subclass should store a reference to the current memory zone.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

NSCoder.h

## systemVersion

During encoding, this method should return the system version currently in effect.

```
- (unsigned)systemVersion
```

### Discussion

During decoding, this method should return the version that was in effect when the data was encoded.

By default, this method returns the current system version, which is appropriate for encoding but not for decoding. Subclasses that implement decoding must override this method to return the system version of the data being decoded.

### Availability

Available in Mac OS X v10.0 and later.

**Declared In**

NSCoder.h

**versionForClassName:**

Returns the version in effect for the class with a given name.

```
- (NSInteger)versionForClassName:(NSString *)className
```

**Return Value**

The version in effect for the class named *className* or `NSNotFound` if no class named *className* exists.

**Discussion**

When encoding, this method returns the current version number of the class. When decoding, this method returns the version number of the class being decoded. Subclasses must override this method.

**Special Considerations**

The version number applies to `NSArchiver/NSUnarchiver`, but not to `NSKeyedArchiver/NSKeyedUnarchiver`. A keyed archiver does not encode class version numbers.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

```
+ setVersion:(NSObject)
```

```
+ version(NSObject)
```

**Declared In**

NSCoder.h



# Deprecated NSCoder Methods

---

A method identified as deprecated has been superseded and may become unsupported in the future.

## Deprecated in Mac OS X v10.5

### **decodeNXObject**

Decodes an object previously written with `encodeNXObject:` (page 33). (Deprecated in Mac OS X v10.5.)

- (id)decodeNXObject

#### **Discussion**

No sharing is done across separate `decodeNXObject` invocations. Callers must have implemented an `initWithCoder:`, which parallels the `read:` methods, on all of their classes that may be touched by this operation. The returned object is autoreleased.

#### **Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

#### **Declared In**

`NSCoder.h`

### **encodeNXObject:**

Encodes an old-style object onto the coder. (Deprecated in Mac OS X v10.5.)

- (void)encodeNXObject:(id)object

#### **Discussion**

No sharing is done across separate `encodeNXObject:` invocations. Callers must have implemented an `encodeWithCoder:`, which parallels the `write:` methods, on all of their classes that may be touched by this operation.

#### **Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

#### **Declared In**

`NSCoder.h`



# Document Revision History

---

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

Date	Notes
2006-07-23	Updated for Mac OS X v10.5.
2006-06-28	Enhanced the description of <code>versionForClassName:</code> .
2006-05-23	First publication of this content as a separate document.

**REVISION HISTORY**

Document Revision History

# Index

---

## A

---

`allowsKeyedCoding` [instance method 9](#)

## C

---

`containsValueForKey:` [instance method 10](#)

## D

---

`decodeArrayOfObjCType:count:at:` [instance method 10](#)

`decodeBoolForKey:` [instance method 10](#)

`decodeBytesForKey:returnedLength:` [instance method 11](#)

`decodeBytesWithReturnedLength:` [instance method 11](#)

`decodeDataObject` [instance method 12](#)

`decodeDoubleForKey:` [instance method 12](#)

`decodeFloatForKey:` [instance method 12](#)

`decodeInt32ForKey:` [instance method 13](#)

`decodeInt64ForKey:` [instance method 13](#)

`decodeIntegerForKey:` [instance method 14](#)

`decodeIntForKey:` [instance method 14](#)

`decodeNXObject` [instance method 33](#)

`decodeObject` [instance method 14](#)

`decodeObjectForKey:` [instance method 15](#)

`decodePoint` [instance method 15](#)

`decodePointForKey:` [instance method 16](#)

`decodePropertyList` [instance method 16](#)

`decodeRect` [instance method 16](#)

`decodeRectForKey:` [instance method 16](#)

`decodeSize` [instance method 17](#)

`decodeSizeForKey:` [instance method 17](#)

`decodeValueOfObjCType:at:` [instance method 17](#)

`decodeValuesOfObjCTypes:` [instance method 18](#)

## E

---

`encodeArrayOfObjCType:count:at:` [instance method 18](#)

`encodeBool:forKey:` [instance method 19](#)

`encodeBycopyObject:` [instance method 19](#)

`encodeByrefObject:` [instance method 20](#)

`encodeBytes:length:` [instance method 20](#)

`encodeBytes:length:forKey:` [instance method 21](#)

`encodeConditionalObject:` [instance method 21](#)

`encodeConditionalObject:forKey:` [instance method 21](#)

`encodeDataObject:` [instance method 22](#)

`encodeDouble:forKey:` [instance method 22](#)

`encodeFloat:forKey:` [instance method 23](#)

`encodeInt32:forKey:` [instance method 23](#)

`encodeInt64:forKey:` [instance method 24](#)

`encodeInt:forKey:` [instance method 24](#)

`encodeInteger:forKey:` [instance method 24](#)

`encodeNXObject:` [instance method 33](#)

`encodeObject:` [instance method 25](#)

`encodeObject:forKey:` [instance method 25](#)

`encodePoint:` [instance method 26](#)

`encodePoint:forKey:` [instance method 26](#)

`encodePropertyList:` [instance method 27](#)

`encodeRect:` [instance method 27](#)

`encodeRect:forKey:` [instance method 27](#)

`encodeRootObject:` [instance method 27](#)

`encodeSize:` [instance method 28](#)

`encodeSize:forKey:` [instance method 28](#)

`encodeValueOfObjCType:at:` [instance method 29](#)

`encodeValuesOfObjCTypes:` [instance method 29](#)

## O

---

`objectZone` [instance method 30](#)

## S

---

setObjectZone: [instance method 30](#)  
systemVersion [instance method 30](#)

## V

---

versionForClassName: [instance method 31](#)