Class

NSCoder

An abstract class that serves as the basis for objects that enable archiving and distribution of other objects.

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.0+ | macOS 10.0+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
class NSCoder
```

Overview

`NSCoder` declares the interface used by concrete subclasses to transfer objects and other values between memory and some other format. This capability provides the basis for archiving (storing objects and data on disk) and distribution (copying objects and data items 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 "coder classes", and instances of these classes are "coder objects" (or simply "coders"). A coder that can only encode values is an "encoder", and one that can only decode values is a "decoder".

`NSCoder` operates on objects, scalars, C arrays, structures, strings, and on pointers to these types. It doesn't handle types whose implementation varies across platforms, such as `union`, `void *`, function pointers, and long chains of pointers. A coder 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.

The AVFoundation framework adds methods to the `NSCoder` class to make it easier to create archives including Core Media time structures, and extract Core Media time structure from archives.

Subclassing Notes

For details of how to create a subclass of `NSCoder`, see Subclassing NSCoder in Archives and Serializations Programming Guide.

Topics

Inspecting a Coder

`var allowsKeyedCoding: Bool`

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

`func containsValue(forKey: String) -> Bool`

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

`var decodingFailurePolicy: NSCoder.DecodingFailurePolicy`

The action the coder should take when decoding fails.

`enum DecodingFailurePolicy`

Policies describing the action the coder should take when encountering decode failures.

Encoding General Data

`func encodeArray(ofObjCType: UnsafePointer<CChar>, count: Int, at: UnsafeRawPointer)`

Encodes an array of the given Objective-C type, provided the number of items and a pointer.

`func encode(Bool, forKey: String)`

Encodes a Boolean value and associates it with the string `key`.

`func encodeBycopyObject(Any?)`

An encoding method for subclasses to override such that it creates a copy, rather than a proxy, when decoded.

`func encodeByrefObject(Any?)`

An encoding method for subclasses to override such that it creates a proxy, rather than a copy, when decoded.

`func encodeBytes(UnsafeRawPointer?, length: Int)`

Encodes a buffer of data of an unspecified type.

`func encodeBytes(UnsafePointer<UInt8>?, length: Int, forKey: String)`

Encodes a buffer of data, given its length and a pointer, and associates it with a string key.

`func encodeConditionalObject(Any?)`

An encoding method for subclasses to override to conditionally encode an object, preserving common references to it.

`func encodeConditionalObject(Any?, forKey: String)`

An encoding method for subclasses to override to conditionally encode an object, preserving common references to it, only if it has been unconditionally encoded.

`func encode(Data)`

Encodes a given data object.

`func encode(Double, forKey: String)`

Encodes a double-precision floating point value and associates it with the string key.

`func encode(Float, forKey: String)`

Encodes a floating point value and associates it with the string key.

`func encodeCInt(Int32, forKey: String)`

Encodes a C integer value and associates it with the string key.

`func encode(Int, forKey: String)`

Encodes an integer value and associates it with the string key.

`func encode(Int32, forKey: String)`

Encodes a 32-bit integer value and associates it with the string key.

`func encode(Int64, forKey: String)`

Encodes a 64-bit integer value and associates it with the string key.

`func encode(Any?)`

Encodes an object.

`func encode(Any?, forKey: String)`

Encodes an object and associates it with the string key.

`func encode(NSPoint)`

Encodes a point.

`func encode(NSPoint, forKey: String)`

Encodes a point and associates it with the string key.

`func encodePropertyList(Any)`

Encodes a property list.

`func encode(NSRect)`

Encodes a rectangle structure.

`func encode(NSRect, forKey: String)`

Encodes a rectangle structure and associates it with the string key.

`func encodeRootObject(Any)`

An encoding method for subclasses to override to encode an interconnected group of objects, starting with the provided root object.

`func encode(NSSize)`

Encodes a size structure.

`func encode(NSSize, forKey: String)`

Encodes a size structure and associates it with the given string key.

`func encodeValue(ofObjCType: UnsafePointer<CChar>, at: UnsafeRawPointer)`

Encodes a value of the given type at the given address.

Encoding Geometry-Based Data

`func encode(CGAffineTransform, forKey: String)`

Encodes an affine transform and associates it with the specified key in the receiver's archive.

`func encode(CGPoint, forKey: String)`

Encodes a point and associates it with the specified key in the receiver's archive.

`func encode(CGRect, forKey: String)`

Encodes a rectangle and associates it with the specified key in the receiver's archive.

`func encode(CGSize, forKey: String)`

Encodes size information and associates it with the specified key in the coder's archive.

`func encode(CGVector, forKey: String)`

Encodes vector data and associates it with the specified key in the coder's archive.

`func encode(NSDirectionalEdgeInsets, forKey: String)`

Encodes directional edge inset data and associates it with the specified key in the coder's archive.

`func encode(UIEdgeInsets, forKey: String)`

Encodes edge inset data and associates it with the specified key in the coder's archive.

`func encode(UIOffset, forKey: String)`

Encodes offset data and associates it with the specified key in the coder's archive.

Encoding Core Media Time Structures

`func encode(CMTime, forKey: String)`

Encodes a given Core Media time structure and associates it with a specified key.

`func encode(CMTimeRange, forKey: String)`

Encodes a given Core Media time range structure and associates it with a specified key.

`func encode(CMTimeMapping, forKey: String)`

Encodes a given Core Media time mapping structure and associates it with a specified key.

Secure Coding

`var requiresSecureCoding: Bool`

Indicates whether the archiver requires all archived classes to resist object substitution attacks.

`var allowedClasses: Set<AnyHashable>?`

The set of coded classes allowed for secure coding.

Decoding Top-Level Objects

`func decodeObject<DecodedObjectType>(of: DecodedObjectType.Type, forKey: String) -> DecodedObjectType?`

Decode an object as an expected type, failing if the archived type doesn't match.

`func decodeObject(of: [AnyClass]?, forKey: String) -> Any?`

Decode an object as one of several expected types, failing if the archived type doesn't match any of the types.

~~`func decodeTopLevelObject() throws -> Any?`~~

Decodes a previously-encoded object.

Deprecated

`func decodeTopLevelObject(forKey: String) throws -> Any?`

Decodes the previously-encoded object associated by a key.

`func decodeTopLevelObject<DecodedObjectType>(of: DecodedObjectType.Type, forKey: String) throws -> DecodedObjectType?`

Decode an object as one of several expected types, failing if the archived type does not match.

`func decodeTopLevelObject(of: [AnyClass]?, forKey: String) throws -> Any?`

Decode an object as one of several expected types, failing if the archived type does not match.

Decoding General Data

`func decodeArray(ofObjCType: UnsafePointer<CChar>, count: Int, at: UnsafeMutableRawPointer)`

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

`func decodeBool(forKey: String) -> Bool`

Decodes and returns a boolean value that was previously encoded with `encode(_:forKey:)` and associated with the string `key`.

`func decodeBytes(forKey: String, returnedLength: UnsafeMutablePointer<Int>?) -> UnsafePointer<UInt8>?`

Decodes a buffer of data that was previously encoded with `encodeBytes(_:length:forKey:)` and associated with the string `key`.

`func decodeBytes(withReturnedLength: UnsafeMutablePointer<Int>) -> UnsafeMutableRawPointer?`

Decodes a buffer of data whose types are unspecified.

`func decodeData() -> Data?`

Decodes and returns an `NSData` object that was previously encoded with `encode(_:)`. Subclasses must override this method.

`func decodeDouble(forKey: String) -> Double`

Decodes and returns a double value that was previously encoded with either encode(_:forKey:) or encode(_:forKey:) and associated with the string key.

```
func decodeFloat(forKey: String) -> Float
```

Decodes and returns a float value that was previously encoded with encode(_:forKey:) or encode(_:forKey:) and associated with the string key.

```
func decodeCInt(forKey: String) -> Int32
```

Decodes and returns an int value that was previously encoded with encodeCInt(_:forKey:), encode(_:forKey:), encode(_:forKey:), or encode(_:forKey:) and associated with the string key.

```
func decodeInteger(forKey: String) -> Int
```

Decodes and returns an NSInteger value that was previously encoded with encodeCInt(_:forKey:), encode(_:forKey:), encode(_:forKey:), or encode(_:forKey:) and associated with the string key.

```
func decodeInt32(forKey: String) -> Int32
```

Decodes and returns a 32-bit integer value that was previously encoded with encodeCInt(_:forKey:), encode(_:forKey:), encode(_:forKey:), or encode(_:forKey:) and associated with the string key.

```
func decodeInt64(forKey: String) -> Int64
```

Decodes and returns a 64-bit integer value that was previously encoded with encodeCInt(_:forKey:), encode(_:forKey:), encode(_:forKey:), or encode(_:forKey:) and associated with the string key.

```
func decodeObject() -> Any?
```

Decodes and returns an object that was previously encoded with any of the encode…Object methods.

```
func decodeObject(forKey: String) -> Any?
```

Decodes and returns a previously-encoded object that was previously encoded with encode(_:forKey:) or encodeConditionalObject(_:forKey:) and associated with the string key.

```
func decodePoint() -> NSPoint
```

Decodes and returns an NSPoint structure that was previously encoded with encode(_:).

```
func decodePoint(forKey: String) -> NSPoint
```

Decodes and returns an NSPoint structure that was previously encoded with encode(_:forKey:).

`func decodePropertyList() -> Any?`

Decodes a property list that was previously encoded with `encodePropertyList(_:)`.

`func decodeRect() -> NSRect`

Decodes and returns an NSRect structure that was previously encoded with `encode(_:)`.

`func decodeRect(forKey: String) -> NSRect`

Decodes and returns an NSRect structure that was previously encoded with `encode(_:forKey:)`.

`func decodeSize() -> NSSize`

Decodes and returns an NSSize structure that was previously encoded with `encode(_:)`.

`func decodeSize(forKey: String) -> NSSize`

Decodes and returns an NSSize structure that was previously encoded with `encode(_:forKey:)`.

~~`func decodeValue(ofObjCType: UnsafePointer<CChar>, at: UnsafeMutableRawPointer)`~~

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

Deprecated

`func decodeValue(ofObjCType: UnsafePointer<CChar>, at: UnsafeMutableRawPointer, size: Int)`

Decodes a single value of a known type from the specified data buffer.

`func decodePropertyList(forKey: String) -> Any?`

Returns a decoded property list for the specified key.

Decoding Geometry-Based Data

`func decodeCGAffineTransform(forKey: String) -> CGAffineTransform`

Decodes and returns the Core Graphics affine transform structure associated with the specified key in the coder's archive.

`func decodeCGPoint(forKey: String) -> CGPoint`

Decodes and returns the Core Graphics point structure associated with the specified key in the coder's archive.

`func decodeCGRect(forKey: String) -> CGRect`

Decodes and returns the Core Graphics rectangle structure associated with the specified key in the coder's archive.

`func decodeCGSize(forKey: String) -> CGSize`

Decodes and returns the Core Graphics size structure associated with the specified key in the coder's archive.

`func decodeCGVector(forKey: String) -> CGVector`

Decodes and returns the Core Graphics vector data associated with the specified key in the coder's archive.

`func decodeDirectionalEdgeInsets(forKey: String) -> NSDirectionalEdgeInsets`

Decodes and returns the UIKit directional edge insets structure associated with the specified key in the coder's archive.

`func decodeUIEdgeInsets(forKey: String) -> UIEdgeInsets`

Decodes and returns the UIKit edge insets structure associated with the specified key in the coder's archive.

`func decodeUIOffset(forKey: String) -> UIOffset`

Decodes and returns the UIKit offset structure associated with the specified key in the coder's archive.

Decoding Core Media Time Structures

`func decodeTime(forKey: String) -> CMTime`

Returns the Core Media time structure associated with a given key.

`func decodeTimeRange(forKey: String) -> CMTimeRange`

Returns the Core Media time range structure associated with a given key.

`func decodeTimeMapping(forKey: String) -> CMTimeMapping`

Returns the Core Media time mapping structure associated with a given key.

Managing Decode Errors

`func failWithError(any Error)`

Signals to this coder that the decode operation has failed.

`var error: (any Error)?`

An error in the top-level encode.

Getting Version Information

`var systemVersion: UInt32`

The system version in effect for the archive.

`func version(forClassName: String) -> Int`

This method is present for historical reasons and is not used with keyed archivers.

Representing Geometric Types as Strings

Convenience methods for creating encodable and decodable types in Objective-C.

`class func cgAffineTransform(for: String) -> CGAffineTransform`

Returns a Core Graphics affine transform structure corresponding to the data in a given string.

`class func cgPoint(for: String) -> CGPoint`

Returns a Core Graphics point structure corresponding to the data in a given string.

`class func cgRect(for: String) -> CGRect`

Returns a Core Graphics rectangle structure corresponding to the data in a given string.

`class func cgSize(for: String) -> CGSize`

Returns a Core Graphics size structure corresponding to the data in a given string.

`class func cgVector(for: String) -> CGVector`

Returns a Core Graphics vector corresponding to the data in a given string.

`class func nsDirectionalEdgeInsets(for: String) -> NSDirectionalEdgeInsets`

Returns a directional edge insets structure based on data in the specified string.

`class func uiEdgeInsets(for: String) -> UIEdgeInsets`

Returns a UIKit edge insets structure based on the data in the specified string.

`class func uiOffset(for: String) -> UIOffset`

Returns a UIKit offset structure corresponding to the data in a given string.

`class func string(for: CGRect) -> String`

Returns a string formatted to contain the data from a rectangle.

`class func string(for: CGVector) -> String`

Returns a string formatted to contain the data from a vector data structure.

`class func string(for: CGAffineTransform) -> String`

Returns a string formatted to contain the data from an affine transform.

`class func string(for: CGPoint) -> String`

Returns a string formatted to contain the data from a point.

`class func string(for: CGSize) -> String`

Returns a string formatted to contain the data from a size data structure.

`class func string(for: NSDirectionalEdgeInsets) -> String`

Returns a string formatted to contain the data from a directional edge insets structure.

`class func string(for: UIEdgeInsets) -> String`

Returns a string formatted to contain the data from an edge insets structure.

`class func string(for: UIOffset) -> String`

Returns a string formatted to contain the data from an offset structure.

Error Codes

`var NSCoderErrorMaximum: Int`

The end of the range of error codes reserved for coder errors.

`var NSCoderErrorMinimum: Int`

The start of the range of error codes reserved for coder errors.

`var NSCoderReadCorruptError: Int`

Decoding failed due to corrupt data.

`var NSCoderValueNotFoundError: Int`

The requested data wasn't found.

`var NSCoderInvalidValueError: Int`

Data wasn't valid to encode.

Instance Methods

```
func decodeArrayOfObjects<DecodedObject>(ofClass: DecodedObject.Type, forKey: String) -> [DecodedObject]?
```

```
func decodeArrayOfObjects(ofClasses: [AnyClass], forKey: String) -> [Any]?
```

```
func decodeBytes(forKey: String, minimumLength: Int) -> UnsafePointer<UInt8>?
```

Decode bytes from the decoder for a given key. The length of the bytes must be greater than or equal to the `length` parameter. If the result exists, but is of insufficient length, then the decoder uses `failWithError` to fail the entire decode operation. The result of that is configurable on a per-NSCoder basis using `NSDecodingFailurePolicy`.

```
func decodeBytes(withMinimumLength: Int) -> UnsafeMutableRawPointer?
```

Decode bytes from the decoder. The length of the bytes must be greater than or equal to the `length` parameter. If the result exists, but is of insufficient length, then the decoder uses `failWithError` to fail the entire decode operation. The result of that is configurable on a per-NSCoder basis using `NSDecodingFailurePolicy`.

```
func decodeDictionary<DecodedKey, DecodedObject>(withKeyClass: DecodedKey.Type, objectClass: DecodedObject.Type, forKey: String) -> [DecodedKey : DecodedObject]?
```

```
func decodeDictionary(withKeysOfClasses: [AnyClass], objectsOfClasses: [AnyClass], forKey: String) -> [AnyHashable : Any]?
```

~~`func decodeTopLevelObject(forKey: String) throws -> AnyObject?`~~ Deprecated

Relationships

Inherits From

`NSObject`

Inherited By

`NSArchiver`
`NSKeyedArchiver`
`NSKeyedUnarchiver`
`NSUnarchiver`
`NSXPCCoder`

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSObjectProtocol

See Also

Keyed Archivers

`class` NSKeyedArchiver

An encoder that stores an object's data to an archive referenced by keys.

`protocol` NSKeyedArchiverDelegate

The optional methods implemented by the delegate of a keyed archiver.

`class` NSKeyedUnarchiver

A decoder that restores data from an archive referenced by keys.

`protocol` NSKeyedUnarchiverDelegate

The optional methods implemented by the delegate of a keyed unarchiver.

`class` NSSecureUnarchiveFromDataTransformer

A value transformer that converts data to and from classes that support secure coding.