Protocol

# NSSecureCoding

**A protocol that enables encoding and decoding in a manner that is robust against object substitution attacks.**

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

```
protocol NSSecureCoding : NSCoding
```

## Overview

Historically, many classes decoded instances of themselves like this:

Swift | Objective-C

```
if let object = decoder.decodeObjectForKey("myKey") as MyClass {
    // ...succeeds...
} else {
    // ...fail...
}
```

This technique is potentially unsafe because by the time you can verify the class type, the object has already been constructed, and if this is part of a collection class, potentially inserted into an object graph.

In order to conform to `NSSecureCoding`:

- An object that does not override doc://com.apple.documentation/documentation/oslog/oslogentry/init(coder:) can conform to

`NSSecureCoding` without any changes (assuming that it is a subclass of another class that conforms).

- An object that does override doc://com.apple.documentation/documentation/oslog/oslogentry/init(coder:) must decode any enclosed objects using the `decodeObjectOfClass:forKey:` method. For example:

Swift Objective-C

```
let obj = decoder.decodeObject(of:MyClass.self, forKey: "myKey")
```

In addition, the class must override the getter for its `supportsSecureCoding` property to return `true`.

For more information about how this relates to the NSXPC API, see Creating XPC Services in Daemons and Services Programming Guide.

# Topics

## Checking for Secure Coding

`static var supportsSecureCoding: Bool`

A Boolean value that indicates whether or not the class supports secure coding.

**Required**

# Relationships

## Inherits From

`NSCoding`

## Conforming Types

`CachedURLResponse`
`Dimension`
`FileHandle`
`FileWrapper`

HTTPURLResponse
ISO8601DateFormatter
MeasurementFormatter
NSAffineTransform
NSAppleEventDescriptor
NSArray
NSAttributedString
NSCalendar
NSCharacterSet
NSComparisonPredicate
NSCompoundPredicate
NSCountedSet
NSData
NSDataDetector
NSDate
NSDateComponents
NSDateInterval
NSDecimalNumber
NSDictionary
NSError
NSException
NSExpression
NSExtensionItem
NSFileSecurity
NSHashTable
NSIndexPath
NSIndexSet
NSLocale
NSMapTable
NSMeasurement
NSMutableArray
NSMutableAttributedString
NSMutableCharacterSet
NSMutableData
NSMutableDictionary
NSMutableIndexSet
NSMutableOrderedSet
NSMutableSet
NSMutableString
NSMutableURLRequest
NSNull
NSNumber
NSOrderedSet

NSOrthography
NSPersonNameComponents
NSPointerArray
NSPredicate
NSPurgeableData
NSRegularExpression
NSSet
NSSortDescriptor
NSString
NSTextCheckingResult
NSTimeZone
NSURL
NSURLQueryItem
NSURLRequest
NSUUID
NSValue
NSXPCListenerEndpoint
URLAuthenticationChallenge
URLCredential
URLProtectionSpace
URLResponse
Unit
UnitAcceleration
UnitAngle
UnitArea
UnitConcentrationMass
UnitConverterLinear
UnitDispersion
UnitDuration
UnitElectricCharge
UnitElectricCurrent
UnitElectricPotentialDifference
UnitElectricResistance
UnitEnergy
UnitEnergy.EnergyKit
UnitFrequency
UnitFuelEfficiency
UnitIlluminance
UnitInformationStorage
UnitLength
UnitMass
UnitPower
UnitPressure

`UnitSpeed`
`UnitTemperature`
`UnitVolume`

---

# See Also

## Adopting Codability

Encoding and Decoding Custom Types

Make your data types encodable and decodable for compatibility with external representations such as JSON.

`typealias Codable = Decodable & Encodable`

A type that can convert itself into and out of an external representation.

`protocol NSCoding`

A protocol that enables an object to be encoded and decoded for archiving and distribution.