Class

NSKeyedUnarchiver

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

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

```
class NSKeyedUnarchiver
```

Overview

`NSKeyedUnarchiver` is a concrete subclass of `NSCoder` that defines methods for decoding a set of named objects (and scalar values) from a keyed archive. The `NSKeyedArchiver` class produces archives that this class can decode.

The archiver creates keyed archive as a hierarchy of objects. The archiver treats each object as a namespace into which it can encode other objects. This means that an unarchiver can only decode objects encoded within the immediate scope of their parent object. Objects encoded elsewhere in the hierarchy — whether higher than, lower than, or parallel to this particular object — aren't 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`-prefixed methods of this class using a key that does not exist in the archive, the return value indicates failure. This value varies by decoded type. For example, if a key does not exist in an archive, `decodeBool(forKey:)` returns `false`, `decodeIntForKey:` returns `0`, and `decodeObject(forKey:)` returns `nil`.

`NSKeyedUnarchiver` supports limited type coercion for numeric types. You can use any of the integer decode methods to decode a value encoded as any type of integer, whether a standard `Int` or an explicit 32-bit or 64-bit integer. Likewise, you can use the `Float`- or `Double`-returning decode methods to handle value encoded as a `Float` or `Double`. If an encoded value is too large to fit within the coerced type, the decoding method throws a `rangeException`. Further, when

trying to coerce a value to an incompatible type — for example decoding an `Int` as a `Float` — the decoding method throws an `invalidUnarchiveOperationException`.

Topics

Creating a Keyed Unarchiver

`init(forReadingFrom: Data) throws`

Initializes an archiver to decode data from the specified location.

~~`init()`~~

Initializes an archiver to decode data.

Deprecated

~~`init(forReadingWith: Data)`~~

Initializes an archiver to decode data from the specified location.

Deprecated

Unarchiving Data

`class func unarchiveTopLevelObjectWithData(Data) throws -> Any?`

Decodes a previously-archived object graph, and returns the root object.

`static func unarchivedObject<DecodedObjectType>(ofClass: DecodedObjectType.Type, from: Data) throws -> DecodedObjectType?`

Decodes a previously-archived object graph, and returns the root object as the specified type.

`class func unarchivedObject(ofClasses: Set<AnyHashable>, from: Data) throws -> Any`

Decodes a previously-archived object graph, returning the root object as one of the specified classes.

`static func unarchivedObject(ofClasses: [AnyClass], from: Data) throws -> Any?`

Decodes a previously-archived object graph, returning the root object as one of the specified classes.

`var requiresSecureCoding: Bool`

Indicates whether the receiver requires all unarchived classes to conform to NSSecureCoding.

~~class func unarchiveObject(with: Data) -> Any?~~

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

Deprecated

~~class func unarchiveObject(withFile: String) -> Any?~~

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

Deprecated

Decoding Data

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

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

```
func decodeDecodable<T>(T.Type, forKey: String) -> T?
```

Decodes a decodable value associated with a given key.

```
func decodeTopLevelDecodable<T>(T.Type, forKey: String) throws -> T?
```

Decodes a top-level decodable value associated with a given key.

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

Decodes a Boolean value associated with a given key.

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

Decodes a stream of bytes associated with a given key.

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

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

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

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

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

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

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

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

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

Decodes and returns an object associated with a given key.

```
func finishDecoding()
```

Tells the receiver that you are finished decoding objects.

```
var decodingFailurePolicy: NSCoder.DecodingFailurePolicy
```

The action to take when this unarchiver fails to decode an entry.

Managing the Delegate

```
var delegate: (any NSKeyedUnarchiverDelegate)?
```

The receiver's delegate.

Managing Class Names

```
class func setClass(AnyClass?, forClassName: String)
```

Sets a global translation mapping to decode objects encoded with a given class name as instances of a given class instead.

```
class func `class`(forClassName: String) -> AnyClass?
```

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

```
func setClass(AnyClass?, forClassName: String)
```

Sets a translation mapping on this unarchiver to decode objects encoded with a given class name as instances of a given class instead.

```
func `class`(forClassName: String) -> AnyClass?
```

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

Constants

Keyed Unarchiving Exception Names

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

Type Methods

```
class func unarchiveTopLevelObjectWithData(NSData) throws -> AnyObject?
```

```
static func unarchivedArrayOfObjects<DecodedObject>(ofClass: Decoded
Object.Type, from: Data) throws -> [DecodedObject]?
```

```
static func unarchivedArrayOfObjects(ofClasses: [AnyClass], from: Data)
throws -> [Any]?
```

```
static func unarchivedDictionary(keysOfClasses: [AnyClass], objectsOf
Classes: [AnyClass], from: Data) throws -> [AnyHashable : Any]?
```

```
static func unarchivedDictionary<DecodedKey, DecodedObject>(ofKeyClass:
DecodedKey.Type, objectClass: DecodedObject.Type, from: Data) throws ->
[DecodedKey : DecodedObject]?
```

Relationships

Inherits From

NSCoder

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSObjectProtocol
Sendable
SendableMetatype

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.

`protocol` NSKeyedUnarchiverDelegate

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

`class` NSCoder

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

`class` NSSecureUnarchiveFromDataTransformer

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