NSUnarchiver

Inherits From:	NSCoder : NSObject
Conforms To:	NSObject (NSObject)
Declared In:	Foundation/NSArchiver.h

Class Description

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. See the NSCoder and NSArchiver specifications for an introduction to archiving.

General Exception Conditions

While unarchiving, NSUnarchiver performs a variety of consistency checks on the incoming data stream. NSUnarchiver raises an NSInconsistentArchiveException when:

- A class name is missing where one is expected.
- A class name is found that refers to an unknown class.
- A type code is found that's different from the one expected.
- An unknown type code is found.
- Excess characters are found in a type code, or characters are missing.

For a description of type codes, see the discussion of the **@encode()** compiler directive in *Object-Oriented Programming and the Objective-C Language*.

Invoking inappropriate methods can also lead to errors. NSUnarchiver's superclass, NSCoder, provides methods for both encoding and decoding. However, only the decoding methods are applicable to NSUnarchiver; don't send an NSUnarchiver any **encode...** messages.

Method Types

Initializing an NSUnarchiver	– initForReadingWithData:
Decoding objects	+ unarchiveObjectWithData: + unarchiveObjectWithFile:

Managing an NSUnarchiver	– isAtEnd
	– objectZone
	– setObjectZone:
	– systemVersion
Substituting classes or objects	+ classNameDecodedForArchiveClassName:
	+ decodeClassName:asClassName:
	– classNameDecodedForArchiveClassName:
	– decodeClassName:asClassName:
	– replaceObject:withObject:

Class Methods

classNameDecodedForArchiveClassName:

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

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

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

See also: – **classNameDecodedForArchiveClassName:**

decodeClassName:asClassName:

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

Instructs instances of NSUnarchiver to use the class named *trueName* when instantiating objects whose ostensible class, according to the archived data, is *nameInArchive*. 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 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.

See also: + **classNameDecodedForArchiveClassName:**, – **decodeClassName:asClassName:**

unarchiveObjectWithData:

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

Decodes and returns the object archived in *data*. This method invokes **initForReadingWithData:** and **decodeObject** to create a temporary NSUnarchiver that decodes the object. If the archived object is the root of a graph of objects, the entire graph is unarchived.

See also: **encodeRootObject:** (NSArchiver)

unarchiveObjectWithFile:

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

Decodes and returns the object archived in the file *path*. This convenience method reads the file by invoking NSData's **dataWithContentsOfFile:** method, and then invokes **unarchiveObjectWithData:**.

Instance Methods

classNameDecodedForArchiveClassName:

– (NSString *)**classNameDecodedForArchiveClassName:**(NSString *)*nameInArchive*

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

See also: + **classNameDecodedForArchiveClassName:**

decodeClassName:asClassName:

– (void)**decodeClassName:**(NSString *)*nameInArchive* **asClassName:**(NSString *)*trueName*

Instructs the receiver to use the class named *trueName* when instantiating objects whose ostensible class, according to the archived data, is *nameInArchive*. 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.

See also: – **classNameDecodedForArchiveClassName:**, + **decodeClassName:asClassName:**

initForReadingWithData:

– (id)**initForReadingWithData:**(NSData *)*data*

Initializes an NSUnarchiver object from the data object *data*, decoding the system version number that was archived in *data* and preparing the NSUnarchiver for a subsequent invocation of **decodeObject**. Raises an NSInvalidArgumentException if *data* is **nil**.

See also: **– systemVersion**

isAtEnd

– (BOOL)**isAtEnd**

Returns YES if the NSUnarchiver has reached the end of the encoded data while decoding, NO if more data follows. 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.

objectZone

– (NSZone *)**objectZone**

Returns the memory zone used to allocate decoded objects.

See also: **– setObjectZone:**

NS+ replaceObject:withObject:

– (void)**replaceObject:**(id)*object* **withObject:**(id)*newObject*

Causes the NSUnarchiver to substitute *newObject* for *object* whenever *object* is extracted from the archive. *newObject* can be of a different class from *object*, and the class mappings set by the two **decodeClassName:asClassName:** methods are ignored.

setObjectZone:

– (void)**setObjectZone:**(NSZone *)*zone*

Set the memory zone used to allocate decoded objects. If *zone* is NULL, or if this method is never invoked, the default zone will be used, as given by **NSDefaultMallocZone()**.

See also: **– objectZone**

systemVersion

– (unsigned int)**systemVersion**

Returns the system version number that was in effect when the archive was created. This information is available as soon as the NSUnarchiver has been initialized.

The version numbers aren’t the usual release designations (such as 2.0 or 3.1). By convention, version numbers under 1000 refer to early versions of NEXTSTEP that didn’t conform to the OpenStep specification.