NSData Class Reference

Data Management: Data Types & Collections

2010-05-11

Contents

NSData Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSMutableCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSData.h
Companion guides	Binary Data Programming Guide Property List Programming Guide
Related sample code	CocoaHTTPServer CocoaSOAP SimplePing Sketch-112 UDPEcho

Overview

`NSData` and its mutable subclass `NSMutableData` provide data objects, object-oriented wrappers for byte buffers. Data objects let simple allocated buffers (that is, data with no embedded pointers) take on the behavior of Foundation objects.

`NSData` creates static data objects, and `NSMutableData` creates dynamic data objects. `NSData` and `NSMutableData` are typically used for data storage and are also useful in Distributed Objects applications, where data contained in data objects can be copied or moved between applications.

Using 32-bit Cocoa, the size of the data is subject to a theoretical 2GB limit (in practice, because memory will be used by other objects this limit will be smaller); using 64-bit Cocoa, the size of the data is subject to a theoretical limit of about 8EB (in practice, the limit should not be a factor).

`NSData` is "toll-free bridged" with its Core Foundation counterpart, *CFData Reference*. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSData *` parameter, you can pass a `CFDataRef`, and in a function where you see a `CFDataRef` parameter, you can pass an `NSData` instance (you cast one type to the other to suppress compiler warnings). This also applies to your concrete subclasses of `NSData`. See Interchangeable Data Types for more information on toll-free bridging.

Adopted Protocols

NSCoding
`encodeWithCoder:`
`initWithCoder:`

NSCopying
`copyWithZone:`

NSMutableCopying
`mutableCopyWithZone:`

Tasks

Creating Data Objects

`+ data` (page 8)
Creates and returns an empty data object.

`+ dataWithBytes:length:` (page 8)
Creates and returns a data object containing a given number of bytes copied from a given buffer.

`+ dataWithBytesNoCopy:length:` (page 9)
Creates and returns a data object that holds *`length`* bytes from the buffer *`bytes`*.

`+ dataWithBytesNoCopy:length:freeWhenDone:` (page 10)
Creates and returns a data object that holds a given number of bytes from a given buffer.

`+ dataWithContentsOfFile:` (page 10)
Creates and returns a data object by reading every byte from the file specified by a given path.

`+ dataWithContentsOfFile:options:error:` (page 11)
Creates and returns a data object by reading every byte from the file specified by a given path.

`+ dataWithContentsOfMappedFile:` (page 12)
Creates and returns a data object from the mapped file specified by *`path`*.

`+ dataWithContentsOfURL:` (page 12)
Returns a data object containing the data from the location specified by a given URL.

`+ dataWithContentsOfURL:options:error:` (page 13)
Creates and returns a data object containing the data from the location specified by *`aURL`*.

`+ dataWithData:` (page 13)
Creates and returns a data object containing the contents of another data object.

`- initWithBytes:length:` (page 16)
Returns a data object initialized by adding to it a given number of bytes of data copied from a given buffer.

`- initWithBytesNoCopy:length:` (page 17)
Returns a data object initialized by adding to it a given number of bytes of data from a given buffer.

`- initWithBytesNoCopy:length:freeWhenDone:` (page 17)
Initializes a newly allocated data object by adding to it *`length`* bytes of data from the buffer *`bytes`*.

- `initWithContentsOfFile:` (page 18)
  Returns a data object initialized by reading into it the data from the file specified by a given path.
- `initWithContentsOfFile:options:error:` (page 18)
  Returns a data object initialized by reading into it the data from the file specified by a given path.
- `initWithContentsOfMappedFile:` (page 19)
  Returns a data object initialized by reading into it the mapped file specified by a given path.
- `initWithContentsOfURL:` (page 19)
  Initializes a newly allocated data object initialized with the data from the location specified by *`aURL`*.
- `initWithContentsOfURL:options:error:` (page 20)
  Returns a data object initialized with the data from the location specified by a given URL.
- `initWithData:` (page 20)
  Returns a data object initialized with the contents of another data object.

Accessing Data

- `bytes` (page 14)
  Returns a pointer to the receiver's contents.
- `description` (page 15)
  Returns an `NSString` object that contains a hexadecimal representation of the receiver's contents.
- `getBytes:length:` (page 15)
  Copies a number of bytes from the start of the receiver's data into a given buffer.
- `getBytes:range:` (page 16)
  Copies a range of bytes from the receiver's data into a given buffer.
- `subdataWithRange:` (page 22)
  Returns a data object containing a copy of the receiver's bytes that fall within the limits specified by a given range.
- `rangeOfData:options:range:` (page 22)
  Finds and returns the range of the first occurrence of the given data, within the given range, subject to given options.
- `getBytes:` (page 29) Deprecated in Mac OS X v10.6
  Copies a data object's contents into a given buffer. (Deprecated. This method is unsafe because it could potentially cause buffer overruns. You should use `getBytes:length:` (page 15) or `getBytes:range:` (page 16) instead.)

Testing Data

- `isEqualToData:` (page 21)
  Compares the receiving data object to *`otherData`*.
- `length` (page 21)
  Returns the number of bytes contained in the receiver.

Storing Data

- `writeToFile:atomically:` (page 23)
  Writes the bytes in the receiver to the file specified by a given path.
- `writeToFile:options:error:` (page 23)
  Writes the bytes in the receiver to the file specified by a given path.
- `writeToURL:atomically:` (page 24)
  Writes the bytes in the receiver to the location specified by *aURL*.
- `writeToURL:options:error:` (page 25)
  Writes the bytes in the receiver to the location specified by a given URL.

Class Methods

data

Creates and returns an empty data object.

```
+ (id)data
```

Return Value
An empty data object.

Discussion
This method is declared primarily for the use of mutable subclasses of `NSData`.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ClipboardViewer
DragNDropOutlineView
QTKitMovieShuffler
StillMotion
With and Without Bindings

Declared In
`NSData.h`

dataWithBytes:length:

Creates and returns a data object containing a given number of bytes copied from a given buffer.

```
+ (id)dataWithBytes:(const void *)bytes length:(NSUInteger)length
```

Parameters

bytes
  A buffer containing data for the new object.

length
: The number of bytes to copy from *bytes*. This value must not exceed the length of *bytes*.

Return Value
A data object containing *length* bytes copied from the buffer *bytes*. Returns `nil` if the data object could not be created.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ dataWithBytesNoCopy:length:` (page 9)
`+ dataWithBytesNoCopy:length:freeWhenDone:` (page 10)

Related Sample Code
CocoaHTTPServer
CocoaSOAP
EnhancedDataBurn
QTCoreVideo301
QTMetadataEditor

Declared In
`NSData.h`

dataWithBytesNoCopy:length:

Creates and returns a data object that holds *length* bytes from the buffer *bytes*.

```
+ (id)dataWithBytesNoCopy:(void *)bytes length:(NSUInteger)length
```

Parameters

bytes
: A buffer containing data for the new object. *bytes* must point to a memory block allocated with `malloc`.

length
: The number of bytes to hold from *bytes*. This value must not exceed the length of *bytes*.

Return Value
A data object that holds *length* bytes from the buffer *bytes*. Returns `nil` if the data object could not be created.

Discussion
The returned object takes ownership of the *bytes* pointer and frees it on deallocation. Therefore, *bytes* must point to a memory block allocated with `malloc`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ dataWithBytes:length:` (page 8)
`+ dataWithBytesNoCopy:length:freeWhenDone:` (page 10)

Declared In
`NSData.h`

dataWithBytesNoCopy:length:freeWhenDone:

Creates and returns a data object that holds a given number of bytes from a given buffer.

```
+ (id)dataWithBytesNoCopy:(void *)bytes length:(NSUInteger)length
    freeWhenDone:(BOOL)freeWhenDone
```

Parameters

`bytes`
A buffer containing data for the new object. If *`freeWhenDone`* is `YES`, *`bytes`* must point to a memory block allocated with `malloc`.

`length`
The number of bytes to hold from *`bytes`*. This value must not exceed the length of *`bytes`*.

`freeWhenDone`
If `YES`, the returned object takes ownership of the *`bytes`* pointer and frees it on deallocation.

Return Value
A data object that holds *`length`* bytes from the buffer *`bytes`*. Returns `nil` if the data object could not be created.

Availability
Available in Mac OS X v10.2 and later.

See Also
`+ dataWithBytes:length:` (page 8)
`+ dataWithBytesNoCopy:length:` (page 9)

Related Sample Code
CocoaSpeechSynthesisExample
PTPPassThrough

Declared In
`NSData.h`

dataWithContentsOfFile:

Creates and returns a data object by reading every byte from the file specified by a given path.

```
+ (id)dataWithContentsOfFile:(NSString *)path
```

Parameters

`path`
The absolute path of the file from which to read data.

Return Value
A data object by reading every byte from the file specified by *`path`*. Returns `nil` if the data object could not be created.

Discussion

This method is equivalent to `dataWithContentsOfFile:options:error:` (page 11) with no options. If you need to know what was the reason for failure, use `dataWithContentsOfFile:options:error:` (page 11).

A sample using this method can be found in "Working With Binary Data".

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ dataWithContentsOfFile:options:error:` (page 11)

`+ dataWithContentsOfMappedFile:` (page 12)

Related Sample Code

CocoaSlides

From A View to A Movie

iSpend

QTMetadataEditor

Reducer

Declared In

`NSData.h`

dataWithContentsOfFile:options:error:

Creates and returns a data object by reading every byte from the file specified by a given path.

```
+ (id)dataWithContentsOfFile:(NSString *)path options:(NSDataReadingOptions)mask
    error:(NSError **)errorPtr
```

Parameters

`path`

The absolute path of the file from which to read data.

`mask`

A mask that specifies options for reading the data. Constant components are described in "`NSDataReadingOptions`" (page 25).

`errorPtr`

If an error occurs, upon return contains an `NSError` object that describes the problem.

Return Value

A data object by reading every byte from the file specified by *`path`*. Returns `nil` if the data object could not be created.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSData.h`

dataWithContentsOfMappedFile:

Creates and returns a data object from the mapped file specified by `path`.

```
+ (id)dataWithContentsOfMappedFile:(NSString *)path
```

Parameters

path

The absolute path of the file from which to read data.

Return Value

A data object from the mapped file specified by *path*. Returns `nil` if the data object could not be created.

Discussion

Because of file mapping restrictions, this method should only be used if the file is guaranteed to exist for the duration of the data object's existence. It is generally safer to use the `dataWithContentsOfFile:` (page 10) method.

This methods assumes mapped files are available from the underlying operating system. A mapped file uses virtual memory techniques to avoid copying pages of the file into memory until they are actually needed.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ dataWithContentsOfFile:` (page 10)

Related Sample Code

FunHouse

Quartz EB

Declared In

`NSData.h`

dataWithContentsOfURL:

Returns a data object containing the data from the location specified by a given URL.

```
+ (id)dataWithContentsOfURL:(NSURL *)aURL
```

Parameters

aURL

The URL from which to read data.

Return Value

A data object containing the data from the location specified by *aURL*. Returns `nil` if the data object could not be created.

Discussion

If you need to know what was the reason for failure, use `dataWithContentsOfURL:options:error:` (page 13).

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ dataWithContentsOfURL:options:error:` (page 13)

`- initWithContentsOfURL:` (page 19)

Related Sample Code

CustomAtomicStoreSubclass

DispatchFractal

LightTable

QTMetadataEditor

WebKitCIPlugIn

Declared In

`NSData.h`

dataWithContentsOfURL:options:error:

Creates and returns a data object containing the data from the location specified by *aURL*.

```
+ (id)dataWithContentsOfURL:(NSURL *)aURL options:(NSDataReadingOptions)mask
    error:(NSError **)errorPtr
```

Parameters

aURL

The URL from which to read data.

mask

A mask that specifies options for reading the data. Constant components are described in "`NSDataReadingOptions`" (page 25).

errorPtr

If there is an error reading in the data, upon return contains an `NSError` object that describes the problem.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- initWithContentsOfURL:` (page 19)

Related Sample Code

ZipBrowser

Declared In

`NSData.h`

dataWithData:

Creates and returns a data object containing the contents of another data object.

```
+ (id)dataWithData:(NSData *)aData
```

Parameters

aData

A data object.

Return Value

A data object containing the contents of *aData*. Returns `nil` if the data object could not be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

`– initWithData:` (page 20)

Related Sample Code

Core Data HTML Store

Declared In

`NSData.h`

Instance Methods

bytes

Returns a pointer to the receiver's contents.

```
- (const void *)bytes
```

Return Value

A read-only pointer to the receiver's contents.

Discussion

If the `length` (page 21) of the receiver is 0, this method returns `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`– description` (page 15)

`– getBytes:` (page 29)

`– getBytes:length:` (page 15)

`– getBytes:range:` (page 16)

Related Sample Code

CocoaHTTPServer

CocoaSOAP

EnhancedDataBurn

UDPEcho

ZipBrowser

Declared In

`NSData.h`

description

Returns an `NSString` object that contains a hexadecimal representation of the receiver's contents.

```
- (NSString *)description
```

Return Value
An `NSString` object that contains a hexadecimal representation of the receiver's contents in `NSData` property list format.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- bytes` (page 14)
- `- getBytes:` (page 29)
- `- getBytes:length:` (page 15)
- `- getBytes:range:` (page 16)

Related Sample Code
Fiendishthngs

Declared In
`NSData.h`

getBytes:length:

Copies a number of bytes from the start of the receiver's data into a given buffer.

```
- (void)getBytes:(void *)buffer length:(NSUInteger)length
```

Parameters

`buffer`
A buffer into which to copy data.

`length`
The number of bytes from the start of the receiver's data to copy to *`buffer`*.

Discussion
The number of bytes copied is the smaller of the *`length`* parameter and the `length` of the data encapsulated in the object.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- bytes` (page 14)
- `- description` (page 15)
- `- getBytes:` (page 29)
- `- getBytes:range:` (page 16)

Related Sample Code
UDPEcho

Declared In
`NSData.h`

getBytes:range:

Copies a range of bytes from the receiver's data into a given buffer.

```
- (void)getBytes:(void *)buffer range:(NSRange)range
```

Parameters

`buffer`
A buffer into which to copy data.

`range`
The range of bytes in the receiver's data to copy to *`buffer`*. The range must lie within the range of bytes of the receiver's data.

Discussion
If *`range`* isn't within the receiver's range of bytes, an `NSRangeException` is raised.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `- bytes` (page 14)
- `- description` (page 15)
- `- getBytes:` (page 29)
- `- getBytes:length:` (page 15)

Declared In
`NSData.h`

initWithBytes:length:

Returns a data object initialized by adding to it a given number of bytes of data copied from a given buffer.

```
- (id)initWithBytes:(const void *)bytes length:(NSUInteger)length
```

Discussion
A data object initialized by adding to it *`length`* bytes of data copied from the buffer *`bytes`*. The returned object might be different than the original receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `+ dataWithBytes:length:` (page 8)
- `- initWithBytesNoCopy:length:` (page 17)
- `- initWithBytesNoCopy:length:freeWhenDone:` (page 17)

Declared In
`NSData.h`

initWithBytesNoCopy:length:

Returns a data object initialized by adding to it a given number of bytes of data from a given buffer.

```
- (id)initWithBytesNoCopy:(void *)bytes length:(NSUInteger)length
```

Parameters

bytes

A buffer containing data for the new object. *bytes* must point to a memory block allocated with `malloc`.

length

The number of bytes to hold from *bytes*. This value must not exceed the length of *bytes*.

Return Value

A data object initialized by adding to it *length* bytes of data from the buffer *bytes*. The returned object might be different than the original receiver.

Discussion

The returned object takes ownership of the *bytes* pointer and frees it on deallocation. Therefore, *bytes* must point to a memory block allocated with `malloc`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ dataWithBytes:length:` (page 8)

`- initWithBytes:length:` (page 16)

`- initWithBytesNoCopy:length:freeWhenDone:` (page 17)

Declared In

`NSData.h`

initWithBytesNoCopy:length:freeWhenDone:

Initializes a newly allocated data object by adding to it *length* bytes of data from the buffer *bytes*.

```
- (id)initWithBytesNoCopy:(void *)bytes length:(NSUInteger)length
    freeWhenDone:(BOOL)flag
```

Parameters

bytes

A buffer containing data for the new object. If *flag* is `YES`, *bytes* must point to a memory block allocated with `malloc`.

length

The number of bytes to hold from *bytes*. This value must not exceed the length of *bytes*.

flag

If `YES`, the returned object takes ownership of the *bytes* pointer and frees it on deallocation.

Availability

Available in Mac OS X v10.2 and later.

See Also

`+ dataWithBytesNoCopy:length:freeWhenDone:` (page 10)

`- initWithBytes:length:` (page 16)
`- initWithBytesNoCopy:length:` (page 17)

Related Sample Code
SonogramViewDemo

Declared In
`NSData.h`

initWithContentsOfFile:

Returns a data object initialized by reading into it the data from the file specified by a given path.

```
- (id)initWithContentsOfFile:(NSString *)path
```

Parameters

path
    The absolute path of the file from which to read data.

Return Value
A data object initialized by reading into it the data from the file specified by *path*. The returned object might be different than the original receiver.

Discussion
This method is equivalent to `initWithContentsOfFile:options:error:` (page 18) with no options.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ dataWithContentsOfFile:` (page 10)
`- initWithContentsOfMappedFile:` (page 19)

Declared In
`NSData.h`

initWithContentsOfFile:options:error:

Returns a data object initialized by reading into it the data from the file specified by a given path.

```
- (id)initWithContentsOfFile:(NSString *)path options:(NSDataReadingOptions)mask
    error:(NSError **)errorPtr
```

Parameters

path
    The absolute path of the file from which to read data.

mask
    A mask that specifies options for reading the data. Constant components are described in "`NSDataReadingOptions`" (page 25).

errorPtr
    If an error occurs, upon return contains an `NSError` object that describes the problem.

Return Value
A data object initialized by reading into it the data from the file specified by *`path`*. The returned object might be different than the original receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ dataWithContentsOfFile:options:error:` (page 11)

Declared In
`NSData.h`

initWithContentsOfMappedFile:

Returns a data object initialized by reading into it the mapped file specified by a given path.

```
- (id)initWithContentsOfMappedFile:(NSString *)path
```

Parameters

`path`
The absolute path of the file from which to read data.

Return Value
A data object initialized by reading into it the mapped file specified by *`path`*. The returned object might be different than the original receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also
`+ dataWithContentsOfMappedFile:` (page 12)
`- initWithContentsOfFile:` (page 18)

Declared In
`NSData.h`

initWithContentsOfURL:

Initializes a newly allocated data object initialized with the data from the location specified by *`aURL`*.

```
- (id)initWithContentsOfURL:(NSURL *)aURL
```

Parameters

`aURL`
The URL from which to read data

Return Value
An `NSData` object initialized with the data from the location specified by *`aURL`*. The returned object might be different than the original receiver.

Availability
Available in Mac OS X v10.0 and later.

See Also

`+ dataWithContentsOfURL:` (page 12)

Declared In

`NSData.h`

initWithContentsOfURL:options:error:

Returns a data object initialized with the data from the location specified by a given URL.

```
- (id)initWithContentsOfURL:(NSURL *)aURL options:(NSDataReadingOptions)mask
    error:(NSError **)errorPtr
```

Parameters

aURL

The URL from which to read data.

mask

A mask that specifies options for reading the data. Constant components are described in "`NSDataReadingOptions`" (page 25).

errorPtr

If there is an error reading in the data, upon return contains an `NSError` object that describes the problem.

Return Value

A data object initialized with the data from the location specified by *aURL*. The returned object might be different than the original receiver.

Availability

Available in Mac OS X v10.4 and later.

See Also

`+ dataWithContentsOfURL:options:error:` (page 13)

Declared In

`NSData.h`

initWithData:

Returns a data object initialized with the contents of another data object.

```
- (id)initWithData:(NSData *)data
```

Parameters

data

A data object.

Return Value

A data object initialized with the contents *data*. The returned object might be different than the original receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also
`+ dataWithData:` (page 13)

Declared In
`NSData.h`

isEqualToData:

Compares the receiving data object to *`otherData`*.

```
- (BOOL)isEqualToData:(NSData *)otherData
```

Parameters

`otherData`
The data object with which to compare the receiver.

Return Value
`YES` if the contents of *`otherData`* are equal to the contents of the receiver, otherwise `NO`.

Discussion
Two data objects are equal if they hold the same number of bytes, and if the bytes at the same position in the objects are the same.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSData.h`

length

Returns the number of bytes contained in the receiver.

```
- (NSUInteger)length
```

Return Value
The number of bytes contained in the receiver.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CocoaSOAP
PTPPassThrough
QTMetadataEditor
UDPEcho
ZipBrowser

Declared In
`NSData.h`

rangeOfData:options:range:

Finds and returns the range of the first occurrence of the given data, within the given range, subject to given options.

```
- (NSRange)rangeOfData:(NSData *)dataToFind options:(NSDataSearchOptions)mask
    range:(NSRange)searchRange
```

Parameters

dataToFind

The data for which to search. This value must not be `nil`.

Important: Raises an `NSInvalidArgumentException` if *dataToFind* is `nil`.

mask

A mask specifying search options. The "NSDataSearchOptions" (page 27) options may be specified singly or by combining them with the C bitwise `OR` operator.

searchRange

The range within the receiver in which to search for *dataToFind*. If this range is not within the receiver's range of bytes, an `NSRangeException` raised.

Return Value

An `NSRange` structure giving the location and length of *dataToFind* within *searchRange*, modulo the options in *mask*. The range returned is relative to the start of the searched data, not the passed-in search range. Returns `{NSNotFound, 0}` if *dataToFind* is not found or is empty `(@"")`.

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSData.h`

subdataWithRange:

Returns a data object containing a copy of the receiver's bytes that fall within the limits specified by a given range.

```
- (NSData *)subdataWithRange:(NSRange)range
```

Parameters

range

The range in the receiver from which to copy bytes. The range must not exceed the bounds of the receiver.

Return Value

A data object containing a copy of the receiver's bytes that fall within the limits specified by *range*.

Discussion

If *range* isn't within the receiver's range of bytes, an `NSRangeException` is raised.

A sample using this method can be found in "Working With Binary Data".

Availability

Available in Mac OS X v10.0 and later.

Declared In
`NSData.h`

writeToFile:atomically:

Writes the bytes in the receiver to the file specified by a given path.

```
- (BOOL)writeToFile:(NSString *)path atomically:(BOOL)flag
```

Parameters

`path`
The location to which to write the receiver's bytes. If *`path`* contains a tilde (~) character, you must expand it with `stringByExpandingTildeInPath` before invoking this method.

`atomically`
If `YES`, the data is written to a backup file, and then—assuming no errors occur—the backup file is renamed to the name specified by *`path`*; otherwise, the data is written directly to *`path`*.

Return Value
`YES` if the operation succeeds, otherwise `NO`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– writeToURL:atomically:` (page 24)

Related Sample Code
CameraBrowser
ImageKitDemo
People
Quartz Composer WWDC 2005 TextEdit
WhackedTV

Declared In
`NSData.h`

writeToFile:options:error:

Writes the bytes in the receiver to the file specified by a given path.

```
- (BOOL)writeToFile:(NSString *)path options:(NSDataWritingOptions)mask
    error:(NSError **)errorPtr
```

Parameters

`path`
The location to which to write the receiver's bytes.

`mask`
A mask that specifies options for writing the data. Constant components are described in "`NSDataWritingOptions`" (page 26).

`errorPtr`
: If there is an error writing out the data, upon return contains an `NSError` object that describes the problem.

Return Value
`YES` if the operation succeeds, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– writeToURL:options:error:` (page 25)

Related Sample Code
From A View to A Movie
From A View to A Picture

Declared In
`NSData.h`

writeToURL:atomically:

Writes the bytes in the receiver to the location specified by *`aURL`*.

```
- (BOOL)writeToURL:(NSURL *)aURL atomically:(BOOL)atomically
```

Parameters

`aURL`
: The location to which to write the receiver's bytes. Only `file://` URLs are supported.

`atomically`
: If `YES`, the data is written to a backup location, and then—assuming no errors occur—the backup location is renamed to the name specified by *`aURL`*; otherwise, the data is written directly to *`aURL`*. *`atomically`* is ignored if *`aURL`* is not of a type the supports atomic writes.

Return Value
`YES` if the operation succeeds, otherwise `NO`.

Discussion
Since at present only `file://` URLs are supported, there is no difference between this method and writeToFile:atomically: (page 23), except for the type of the first argument.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– writeToFile:atomically:` (page 23)

Related Sample Code
AnimatedTableView
Core Data HTML Store
CoreRecipes
CustomAtomicStoreSubclass

Declared In
`NSData.h`

writeToURL:options:error:

Writes the bytes in the receiver to the location specified by a given URL.

```
- (BOOL)writeToURL:(NSURL *)aURL options:(NSDataWritingOptions)mask error:(NSError
    **)errorPtr
```

Parameters

aURL
The location to which to write the receiver's bytes.

mask
A mask that specifies options for writing the data. Constant components are described in "`NSDataWritingOptions`" (page 26).

errorPtr
If there is an error writing out the data, upon return contains an `NSError` object that describes the problem.

Return Value
`YES` if the operation succeeds, otherwise `NO`.

Discussion
Since at present only `file://` URLs are supported, there is no difference between this method and `writeToFile:options:error:` (page 23), except for the type of the first argument.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- writeToFile:options:error:` (page 23)

Declared In
`NSData.h`

Constants

NSDataReadingOptions

Options for methods used to read `NSData` objects.

```
enum {
    NSDataReadingMapped =   1UL << 0,
    NSDataReadingUncached = 1UL << 1
    NSMappedRead = NSDataReadingMapped,
    NSUncachedRead = NSDataReadingUncached
};
typedef NSUInteger NSDataReadingOptions;
```

Constants

`NSDataReadingMapped`

A hint indicating the file should be mapped into virtual memory, if possible.

Available in Mac OS X v10.6 and later.

Declared in `NSData.h`.

`NSDataReadingUncached`

A hint indicating the file should not be stored in the file-system caches.

For data being read once and discarded, this option can improve performance.

Available in Mac OS X v10.6 and later.

Declared in `NSData.h`.

`NSMappedRead`

Deprecated name for `NSDataReadingMapped` (page 26). (Deprecated. Please use `NSDataReadingMapped` (page 26) instead.)

Available in Mac OS X v10.4 and later.

Declared in `NSData.h`.

`NSUncachedRead`

Deprecated name for `NSDataReadingUncached` (page 26). (Deprecated. Please use `NSDataReadingUncached` (page 26) instead.)

Available in Mac OS X v10.4 and later.

Declared in `NSData.h`.

NSDataWritingOptions

Options for methods used to write `NSData` objects.

```
enum {
    NSDataWritingAtomic = 1UL << 0

    NSAtomicWrite = NSDataWritingAtomic
};
typedef NSUInteger NSDataWritingOptions;
```

Constants

`NSDataWritingAtomic`

A hint to write data to an auxiliary file first and then exchange the files. This option is equivalent to using a write method taking the parameter `atomically:YES`.

Available in Mac OS X v10.6 and later.

Declared in `NSData.h`.

`NSAtomicWrite`

Deprecated constant. (Deprecated. Use `NSDataWritingAtomic` (page 26) instead.)

Available in Mac OS X v10.4 and later.

Declared in `NSData.h`.

NSDataSearchOptions

Options for method used to search `NSData` objects. These options are used with the `rangeOfData:options:range:` (page 22) method.

```
enum {
    NSDataSearchBackwards = 1UL << 0,
    NSDataSearchAnchored = 1UL << 1
};
typedef NSUInteger NSDataSearchOptions;
```

Constants

`NSDataSearchBackwards`

Search from the end of `NSData` object.

Available in Mac OS X v10.6 and later.

Declared in `NSData.h`.

`NSDataSearchAnchored`

Search is limited to start (or end, if `NSDataSearchBackwards`) of `NSData` object.

This option performs searching only on bytes at the beginning or end of the range. No match at the beginning or end means nothing is found, even if a matching sequence of bytes occurs elsewhere in the data object.

Available in Mac OS X v10.6 and later.

Declared in `NSData.h`.

Deprecated NSData Methods

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

Deprecated in Mac OS X v10.6

getBytes:

Copies a data object’s contents into a given buffer. (Deprecated in Mac OS X v10.6. This method is unsafe because it could potentially cause buffer overruns. You should use `getBytes:length:` (page 15) or `getBytes:range:` (page 16) instead.)

```
- (void)getBytes:(void *)buffer
```

Parameters

buffer

A buffer into which to copy the receiver's data. The buffer must be at least `length` (page 21) bytes.

Discussion

You can see a sample using this method in “Working With Binary Data”.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `bytes` (page 14)
- `description` (page 15)
- `getBytes:length:` (page 15)
- `getBytes:range:` (page 16)

Related Sample Code

From A View to A Movie

From A View to A Picture

QTCoreVideo301

QTMetadataEditor

Quartz Composer QCTV

Declared In

`NSData.h`

Document Revision History

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

Date	Notes
2010-05-11	Added symbols introduced in iOS 4.0.
2010-01-14	Corrected deprecation information for NSWriteAtomic.
2009-10-13	Cleaned up deprecated enums.
2009-08-28	Correction to search options in rangeOfData:options:range:.
2009-08-09	Corrected typedef declarations. Separated deprecation enums.
2009-05-06	Added subclassing notes. Clarified behavior of bytes and getData:length: methods.
2008-02-08	Corrected typographical errors.
2007-02-27	Updated for Mac OS X v10.5 API.
2006-05-23	First publication of this content as a separate document.