OPENSTEP Release 4.1 Copyright © 1996 by NeXT Software, Inc. All Rights Reserved.

Title: OPENSTEP 4.1 Release Notes: Foundation Framework
Entry Number: 2470
Last Updated: October 21, 1996

OPENSTEP 4.1 Release Notes: Foundation Framework

The Foundation Framework is a library of Objective-C classes providing the infrastructure for non-user-interface object-based applications and other user-interface- and non-user-interface-based frameworks. The OPENSTEP Foundation is compliant with the Foundation portion of the OpenStep specification, and enriches OpenStep with a few extensions.

Within the Foundation are basic classes used by any Objective-C program: collection classes, Distributed Objects, persistence frameworks, Unicode and internationalization support, the foundations of event-driven programming, and facilities to insulate code from the underlying operating system.

These release notes are divided in four main sections:

- Comments on compatibility with OPENSTEP 4.0
- Notes specific to developers who have used previous releases of Foundation

- General comments on often-encountered problem areas and interesting things
- Known problems in this release

In the document below, "4.0" and "OPENSTEP 4.0" refer to all products that shipped with the 4.0 Foundation: OPENSTEP 4.0, D'OLE 4.0, and PDO 4.0 for HP-UX™ and Solaris™. "4.1" and "OPENSTEP 4.1" refers to OPENSTEP 4.1 for Mach and OPENSTEP Enterprise 4.1.

Changes and Compatibility Between OPENSTEP 4.0 and 4.1

There have been a number of changes to Foundation since the shipment of OPENSTEP 4.0, both defect fixes and a few method/function additions. These are enumerated in the section below.

Note that if your application uses API which was added between 4.0 and 4.1, your application might not run under 4.0. (If you do not want to deploy your application on 4.0, then this is not an issue.) You can catch some of the potential problems by defining the macro **STRICT_40** at compile time. This #define is similar to STRICT_OPENSTEP and is used in the Application and Foundation Kits to mark new API. Similarly, if you rely (perhaps unintentionally) on a bug fixed in 4.1, your application may not function correctly under 4.0. Make sure you test on your intended deployment platform.

A number of performance improvements were also made to Foundation between 4.0 and 4.1. Customers developing on 4.1 may find performance to be significantly worse in some cases when a project is deployed on 4.0.

Here are the feature and semantic changes to Foundation in 4.1. A list of most of the bugs fixed follows. Note that a fixed bug may also represent a semantic change from the point of view of the application.

- Some methods in 4.0 were not surrounded by **#if !defined(STRICT_OPENSTEP)** when they weren't, in fact, in OpenStep.

CLASS	HEADER	METHOD
NSCharacterSet	NSCharacterSet.h	+characterSetWithContentsOfFile:
NSConnection	NSConnection.h	-connection:shouldMakeNewConnection:
NSArray	NSPathUtilities.h	-pathsMatchingExtensions:

- These methods have been made public in 4.1. They were not public in 4.0, but existed in the 4.0 Foundation binary so there are no compatibility issues in using them. They are not in OpenStep, however.

CLASS	HEADER	METHOD
NSArray	NSArray.h	-makeObjectsPerformSelector:
NSArray	NSArray.h	-makeObjectsPerformSelector:withObject:
NSSet	NSSet.h	-makeObjectsPerformSelector:
NSSet	NSSet.h	-makeObjectsPerformSelector:withObject:

- These methods and functions were added to Foundation in 4.1 and are not present in the 4.0 Foundation binary. Applications which do not take some precaution in using them will not run correctly on OPENSTEP 4.0. These methods and functions are also not in OpenStep.

CLASS	HEADER	METHOD
NSBundle	NSBundle.h	+allBundles
NSBundle	NSBundle.h	+allFrameworks
NSFileHandle	NSFileHandle.h	+fileHandleWithNullDevice
NSFileHandle	NSFileHandle.h	-initWithNativeHandle:closeOnDealloc:
NSFileHandle	NSFileHandle.h	-initWithFileDescriptor:closeOnDealloc:
NSObject	NSObject.h	+instanceMethodSignatureForSelector:

FUNCTION	HEADER
NSFrameAddress	NSDebug.h
NSReturnAddress	NSDebug.h
NSCountFrames	NSDebug.h

- The NSUserDefaults' _*search* instance variable changed type from NSMutableArray * to NSArray *. This shouldn't be a problem, as applications should not be referencing the instance variables of NSUserDefaults directly.
- NSArray **-componentsJoinedByString:** sends **-description** to each of the array's objects and concatenates the results of that together, rather than using the objects directly. This means that this method will now operate on arrays which are composed of objects that are not NSStrings.
- The number-formatting sequences for NSCalendarDates (%d, %e, %H, %I, %j, %m, %M, %S, %w, %y, %Y) during the construction of a description can now contain formatting flags between the `%' and format code, to the extent that the string methods **+stringWithFormat:**, **-appendFormat:**, etc. understand them. Format flags are not understood during parsing from a format, however. Note that per 70220 [below], NSDateFormatters use their format string both to display the date and to parse a date from a string (which the user entered in a text field, for example), so this modest extension should not be used in the format string of NSDateFormatters. Setting the date formatter to allow natural language may obviate the problem [but may be undesirable for application reasons].
- The default uncaught exception handler now includes the name of the exception and a "backtrace" in the logged message it generates. The backtrace consists only of the return addresses on the execution stack. Additionally, on Windows™, an application exception is raised (via **RaiseException()**), which can be handled by a debugger.
- The *byref* keyword is no longer ignored when transporting NSCalendarDates (as a parameter or return value) over D.O. Thus, a parameter or return value which is declared with *byref* now results in a proxy to the calendar date instance, rather than a real calendar date instance. By default, NSCalendarDates still go over the wire bycopy. NSDates always go over the wire bycopy.

- The NSBundle class method **+pathForResource:ofType:inDirectory:** now first searches a **Resources** subdirectory of the third parameter if such a subdirectory exists. It used to examine only the directory specified in the third parameter.
- NSDate's **-dateWithCalendarFormat:timeZone:** and NSCalendarDate's **-setCalendarFormat:** now have the documented behavior with respect to the first parameter being **nil**. In 4.0, they would simply set the format to **nil**, which resulted in the various **-description** methods returning an empty string.
- NSString **-stringByAppendingPathComponent:** would refuse to append a component that began with `~' in 4.0. This was to prevent the accidental concatenation of "~*username*" onto a path, but `~' is a valid (but perhaps uncommon) first character of a file name, and that behavior was of marginal benefit, so it was eliminated.
- A missing strings file has been added to the Foundation, so in 4.1 error strings returned from the formatters (NSDateFormatter and NSNumberFormatter) will be localized.

This section contains many of the corrected defects to Foundation between OPENSTEP 4.0 and 4.1. Any of these present potential backwards-deployment issues.

Reference: 48447 [FIXED]

Platforms: HP-UX™

Problem: D.O. crashed sending some structures containing doubles

Description: D.O. would crash when attempting to send a parameter or return value which was a structure larger than 8 bytes and contained a double field.

Reference: 50670 [FIXED]

Platforms: All

Problem: NSMethodSignature's **-getArgumentTypeAtIndex:** and **-methodReturnType** returned too much information

Description: These NSMethodSignature methods returned a string which contained the type descriptors for the requested argument plus all the type descriptors for the arguments following.

Reference: 55579 [FIXED]

Platforms: All

Problem: Collections that contain themselves didn't unarchive well

Description: Unarchiving an object graph with a collection which contained itself, or contained an object which archived a reference to the collection, might have caused a program either to crash immediately when the archiving system attempted to retain it, or to crash later when a freed object was sent a message. The problem here was that the collection got retained when the recursive reference was unarchived (which is normal) before it had completed its initialization in **-initWithCoder:**, which can be a problem for objects which keep track of their own retain count (as the collection classes do).

Reference: 62019 [FIXED]

Platforms: All

Problem: Character tables were not Unicode 2.0 compliant

Description: The tables used by NSCharacterSet and shipped as resources with the 4.0 Foundation were compliant with Unicode 1.1. The Unicode 2.0 tables were not yet available at the time of release. The tables in 4.1 are Unicode 2.0 compliant.

Reference: 67201 [FIXED]

Platforms: Windows NT™

Problem: Files mapped with **-initWithContentsOfMappedFile:** could not be removed

Description: On Windows™, a file that a process has mapped cannot be removed; this is a feature of the operating system. However, even after an NSData initialized with **-initWithContentsOfMappedFile:** was deallocated, a program may not have been able to immediately remove the file. The problem was that an internal file-mapping object was autoreleased at the time the data was initialized.

Reference: 68041 [FIXED]

Platforms: All

Problem: NSString's **-stringByStandardizingPath** raised an exception in certain circumstances

Description: When passed an absolute path that ended with .., NSString's -**stringByStandardizingPath** method raised an exception.

Reference: 68223 [FIXED]

Platforms: All

Problem: **-stringByAbbreviatingWithTildeInPath** sometimes incorrectly abbreviated

Description: If the first part of a path matched the user's home directory, '~' would be substituted for that prefix, regardless of whether or not the partial component at the end of the prefix was a complete component or not. For example,

```
[@"/me-local/foo/bar" stringByAbbreviatingWithTildeInPath] => @"~/local/foo/bar"
```

if the example was executed by the "me" user whose account is "/me".

Reference: 68447, 68781 [FIXED]

Platforms: All

Problem: D.O. and archiving type signature checks didn't take structure tags into account

Description: The NeXT compiler does not include structure names in type encoding strings produced from typedefs of structures (for example, @encode(NSRange); instead, the structure tag is given as "?". This is a bug in the compiler, but one that cannot be fixed for backwards compatibility reasons. To achieve compatibility with SunSoft's compiler (which correctly emits all structure tags), a "?" as a structure tag was made to act as a "wildcard", matching anything. This is unlikely to affect user applications, but developers who construct their own type encoding strings should note this, as 4.1 will be more permissive than OPENSTEP 4.0 and NEXTSTEP was, a possible backwards-deployment issue.

Reference: 68790 [FIXED]

Platforms: All

Problem: NSConnectionDidDieNotification not sent in some circumstances

Description: If a client attempted to send a message to a server which had died before the client had processed the port death notification (which can take quite a while to arrive), an NSInvalidSendPortException would be raised rather than the NSConnectionDidDieNotification being sent.

Reference: 69174 [FIXED]

Platforms: All

Problem: Lossy conversion to NSASCIIStringEncoding more lossy than intended

Description: When NSString finds a non-ASCII character during lossy conversion to NSASCIIStringEncoding, NSString converts it to an underscore. In 4.0, when NSString found a non-ASCII character during such conversion, NSString converted the character and all the remaining characters in the string to underscores, even if they were ASCII.

Reference: 69321 [FIXED]

Platforms: Windows NT™

Problem: D.O. leaked every messaged object

Description: On NT, every object that was messaged via D.O. was leaked due to an extraneous retain.

Reference: 69471 [FIXED]

Platforms: All

Problem: NSPipe leaked file descriptors

Description: NSPipes and NSFileHandles do not take ownership of the file descriptor by default. If such an object was released without an explicit close of the descriptors, the descriptors were not recovered. The new methods **-initWithNativeHandle:closeOnDealloc:** and **-initWithFileDescriptor:closeOnDealloc:** allow for the creation of file handles which own the descriptors.

Reference: 69699 [FIXED]

Platforms: Windows NT™

Problem: **NSLog()** doesn't send duplicate output

Description: When running under a debugger, **NSLog()** would send output to the debugger and the event viewer and the standard error handle (which is often not defined for applications, but is for tools and other console-oriented processes). Now when a process is running under a debugger, **NSLog()** only sends its output to the debugger. The normal behavior of **NSLog()** is to send the string to the event viewer and standard error.

Reference: 69926 [FIXED]

Platforms: All

Problem: NSNotificationCenter sometimes sent duplicate notifications

Description: In some situations, NSNotificationCenter sent observers two (or more) copies of a single notification. This only happened to observers which were registered to receive any notification name (perhaps with restriction to particular object).

Reference: 70028 [FIXED]

Platforms: All

Problem: Fault objects could not forward messages

Description: EOFault would cause an unrecognized selector exception rather than causing **-forwardInvocation:** to be invoked on the faulted object when the fault was sent a message that the faulted object did not understand (in other words, an object faulted would not get an opportunity for forward an unknown selector). Faulted objects which wish to forward a message which may cause them to be faulted should override the new NSObject class method **+instanceMethodSignatureForSelector:** to return an NSMethodSignature for the eventual target of the forwarded message. See the subsection *Forwarding Messages in OpenStep* in the *General Comments* section below.

Reference: 70220 [FIXED]

Platforms: All

Problem: NSDateFormatter didn't convert input in the same way as output

Description: NSDateFormatters did not use the output format string to parse user-entered strings, so in most cases strings could not be entered in the same format in which they were displayed. Now a date displayed with "%m/%y" (for example) can also be entered in that format.

Reference: 70323 [FIXED]

Platforms: All

Problem: NSDateFormatter does not allow empty string

Description: Once the user had begun typing in a text field that had a date formatter, the user could not leave the field without entering a valid date; in particular, the user could not leave the field empty. Empty is now valid, and the date formatter produces **nil** for an empty string in its **-getObjectValue:forString:errorDescription:** method.

Reference: 70334 [FIXED]

Platforms: All

Problem: Time zones generated with wrong offset

Description: The **+timeZoneForSecondsFromGMT:** was incorrectly using the negative of the supplied parameter to generate the returned time zone. This also affected the parsing of dates where the time zone was specified with a raw GMT offset, like "-0700", but not where the time zone was a name, such as "GMT-0800" or "PST".

Reference: 70440 [FIXED]

Platforms: All

Problem: NSRunLoop incorrectly computed whether or not there was work to be done

Description: In some situations using background monitoring of file handles, NSRunLoop would not take into account the file handles in deciding whether or not it had any more input sources.

Reference: 70580 [FIXED]

Platforms: All

Problem: NSNumber hashed long long numbers incorrectly

Description: NSNumber hashed long longs by simply casting the value to (unsigned int). This caused some equal numbers to have different hash values, breaking the hash±isEqual: invariant.

Reference: 70612, 70718 [FIXED]

Platforms: All

Problem: NSDecimalNumber inaccuracies

Description: An error in the text from which the division algorithm was taken caused inaccuracies in the less significant end of the result of **-decimalNumberByDividingBy:withBehavior:** (in the fractional part, the ones, and tens digits, depending on the size of the numbers). Also, digits of precision were being lost during initialization on the less significant end of the number, and the decimal point was being incorrectly positioned in the result of **-decimalNumberBySubtracting:withBehavior:**.

Reference: 71006 [FIXED]

Platforms: Windows NT™

Problem: **NSFullUserName()** always returned **nil**

Description: The NETAPI32 DLL was not being loaded when needed. Note that this DLL is not implemented on Windows 95™, so this function will always return **nil** on Windows 95™. Applications should be prepared for a **nil** return value.

Reference: 71288 [FIXED]

Platforms: All

Problem: Problem popping initial autorelease pool

Description: A program could crash if it repeatedly created and released the top autorelease pool.

Notes for Users of Foundation Prior to OPENSTEP 4.0

There were several releases of the Foundation Framework prior to OPENSTEP 4.0. Foundation shipped with:

- NEXTSTEP 3.3
- Enterprise Objects Framework 1.0
- Enterprise Objects Framework 1.1
- Portable Distributed Objects 3.0

- D'OLE 3.5
- OPENSTEP 4.0 Prerelease 1
- OPENSTEP 4.0 Prerelease 2
- PDO 4.0 Prerelease 1

This section describes only API changes that have been made between those releases and OPENSTEP 4.0. For changes since OPENSTEP 4.0, see the section above titled *Changes and Compatibility Between OPENSTEP 4.0 and 4.1*. The semantics of many things have changed, but most not in a way that will affect programs (NEXTSTEP 3.3 Foundation-based programs and EOF 1.1 programs, being older, are more likely to be affected). Unfortunately, at this time there is no single source for information about such changes±refer to the *Foundation Framework Reference* and specific class specifications for the current behavior. This document does not describe extensions to Foundation provided by various NeXT products (for example, WebObjects or EOF).

Foundation is now packaged as a ProjectBuilder framework, and resides in **/NextLibrary/Frameworks/Foundation.framework**. The headers can be found in **/NextLibrary/Frameworks/Foundation.framework/Headers**.

All programs based upon previous version of Foundation should be recompiled (even if no conversion is necessary) to function with this version of Foundation. Compiling against the old static "shlib" (the shared library shipped with NEXTSTEP 3.3, EOF 1.1, and PDO 3.0, and used by WebObjects) is not supported in OPENSTEP 4.0 (without EOF 1.1 installed).

This section is divided into a subsection for each of the previous releases of the Foundation Framework. There are no specific notes for Enterprise Objects Framework 1.0, Portable Distributed Objects 3.0, or WebObjects 1.0 developers; please refer to the section for Enterprise Objects Framework 1.1 developers. The API of the Foundation shipped with Portable Distributed Objects 3.0 and WebObjects 1.0 was identical to that in Enterprise Objects Framework 1.1 (other than extensions made by those products).

Notes Specific to NEXTSTEP 3.3 Developers

Items in the tables below (in the section titled *Tables Showing Foundation API Changes*) that are marked with "(NS33)" are changes made to Foundation between NEXTSTEP 3.3 and OPENSTEP 4.0. Also see the general comments in the section *Notes Specific to Enterprise Objects Framework 1.1 Developers* below. NEXTSTEP 3.3 developers should note the following additions to those comments:

- The header **NXAutoreleaseConnection.h** was not in NEXTSTEP 3.3. Comments referring to this header do not apply to NEXTSTEP 3.3 developers.
- Two additional headers, **NSLock.h** and **NSThread.h**, were not in NEXTSTEP 3.3, but are in OPENSTEP 4.0.

Notes Specific to Enterprise Objects Framework 1.1 Developers

Items in the tables below (in the section titled *Tables Showing Foundation API Changes*) marked with "(EOF11)" are changes made to Foundation between EOF 1.1 and OPENSTEP 4.0.

Some additional changes are not contained in the tables or are of particular note:

- Headers are now imported as **<Foundation/...>**, rather than **<foundation/...>**.
- The structure of imports has changed. The Foundation headers import all ANSI C headers they need and some of the Objective-C interfaces, but none of the operating system interfaces. The set of headers imported by each header has also changed in many places. Where a project was getting a header imported "for free" before, it may now need to import additional headers in order to compile.
- The header **NXAutoreleaseConnection.h** has been removed. Its contents have been moved to **NSCompatibility.h**.
- A **Foundation.h** header has been added, which includes the 15 ANSI C headers, and all Foundation headers except **NSCompatibility.h** and **NSDebug.h**.

- The top-level NSAutoreleasePool is no longer created for automatically±a program needs to create it itself. See the *Foundation Framework Reference* and the NSAutoreleasePool class specification for more information on creating autorelease pools.
- NSDictionary no longer constrains keys to be instances of NSString. Any object that meaningfully responds to **-hash**, **-isEqual:**, and **-copyWithZone:** can be an NSDictionary key.
- The name of the exception variable in the **NS_HANDLER...NS_ENDHANDLER** region has been changed from **exception** to **localException**.
- Some methods that are not in OpenStep, and were not guarded with **#if !defined(STRICT_OPENSTEP)**, now are guarded.
- NSObject no longer conforms to the NSCoding protocol. Instead, the classes that can be archived or distributed, and their subclasses, conform to the protocol explicitly.
- NSBundle's and NSNotificationCenter's instance variables are now declared **@private**.
- The type of the parameter to NSBundle's **+bundleForClass:** method changed from **id** to **Class**.
- The type of the parameter to NSValue's **+valueWithPointer:** method changed from **void *** to **const void ***.
- The typedef **NSStringEncoding** in **NSString.h** has changed from a typedef'd enumeration to a typedef'd scalar.
- The value of the enumeration constant **NSUnicodeStringEncoding** in **NSString.h** has changed from 0 to 10.
- The location of the resources has changed, and can now be found in the Foundation framework directory. If, in a previous release of the Foundation, your application was poking around in the Foundation resources, you should carefully consider why you were doing so, and stop doing it if possible.
- The declarations of the NSAutoreleasePool debugging methods and aids have moved from **NSAutoreleasePool.h** to **NSDebug.h**. Some of the methods are destined for obsolescence, and some new methods have been added.
- The header **NSObjCRuntime.h** now imports the headers **objc/zone.h** and **stdarg.h**.

- Many declarations and definitions were removed from the header **NSUtilities.h** and moved elsewhere, but all the previous material is still defined by importing **NSUtilities.h**. The headers to which things were moved are **NSEnumerator.h**, **NSObjCRuntime.h**, **NSDictionary.h**, and **NSString.h**.

Finally, the Foundation shipped with EOF 1.1 did not contain much of the functionality of the Foundation in OPENSTEP 4.0. These headers, and all the API they declare, were added between EOF 1.1 and OPENSTEP 4.0:

NSAttributedString.h	NSFormatter.h	NSPPL.h
NSByteOrder.h	NSGeometry.h	NSProcessInfo.h
NSCompatibility.h	NSHashTable.h	NSProtocolChecker.h
NSConnection.h	NSHost.h	NSProxy.h
NSDateFormatter.h	NSInvocation.h	NSRunLoop.h
NSDebug.h	NSMapTable.h	NSSerialization.h
NSDecimal.h	NSMethodSignature.h	NSSet.h
NSDecimalNumber.h	NSNotificationQueue.h	NSTask.h
NSDistantObject.h	NSNumberFormatter.h	NSTimer.h
NSDistributedLock.h	NSPort.h	NSUserDefaults.h
NSEnumerator.h	NSPortCoder.h	
NSFileHandle.h	NSPortMessage.h	
NSFileManager.h	NSPortNameServer.h	

The API added by these headers is not included in the tables below.

Notes Specific to D'OLE 3.5 Developers

Items in the tables below (in the section titled *Tables Showing Foundation API Changes*) marked with "(40PR1)" are changes made to Foundation between OPENSTEP 4.0 PR1 and OPENSTEP 4.0. This symbol also marks changes that apply to D'OLE 3.5 developers, with the exceptions noted below. See also the general comments in the section *Notes Specific to OPENSTEP 4.0 PR1 Developers* below.

The following are exceptions to the changes noted between OPENSTEP 4.0 PR1 and OPENSTEP 4.0, for D'OLE 3.5 developers:

- The header **NSPPL.h** isn't in D'OLE 3.5, but is in OPENSTEP 4.0. API changes referring to this header do not apply to D'OLE 3.5 developers.
- These methods were in D'OLE 3.5, but not in OPENSTEP 4.0 PR1:

CLASS	HEADER	METHOD	
NSString	NSPathUtilities.h	+pathWithComponents:	(+NeXT) (40PR1)
NSString	NSPathUtilities.h	-isAbsolutePath	(+NeXT) (40PR1)
NSString	NSPathUtilities.h	-pathComponents	(+NeXT) (40PR1)

Notes Specific to OPENSTEP 4.0 PR1 Developers

Items in the tables below (in the section titled *Tables Showing Foundation API Changes*) marked with "(40PR1)" are changes made to Foundation between OPENSTEP 4.0 PR1 and OPENSTEP 4.0.

Some additional changes are not contained in the tables or are of particular note:

- All applications, frameworks, libraries, bundles, tools, etc. compiled in PR1 should be recompiled.
- The header file **NSAccount.h** has been removed from Foundation. This header was NeXT-specific.
- The header file **NSByteStore.h** has been removed from Foundation (both from OpenStep and from NeXT's implementation).

- The import of the header **NSDebug.h** has been removed from **Foundation.h**; however, it still exists. This header contains features intended for debugging, but not for general use.
- **Foundation.h** now imports the 15 ANSI C headers (**assert.h**, **ctype.h**, **errno.h**, **float.h**, **limits.h**, **locale.h**, **math.h**, **setjmp.h**, **signal.h**, **stdarg.h**, **stddef.h**, **stdio.h**, **stdlib.h**, **string.h**, **time.h**) as a convenience.
- The header file **NSObjCRuntime.h** imported six obsolete headers (**ansi/ansi.h**, **objc/typedstream.h**, **objc/objc-class.h**, **architecture/byte_order.h**, **objc/hashtable.h**, **streams/streams.h**) in PR1. It no longer does so. This change may mean that some of these headers may need to be explicitly imported for code written on OPENSTEP 4.0 PR1 to recompile.
- A **STRICT_OPENSTEP** preprocessor guard now surrounds all API in the headers **NSDecimal.h** and **NSDecimalNumber.h**.
- The macro **FOUNDATION_EXPORT** is now always explicitly defined on Windows™ (that is, it cannot be predefined with the **-D** compiler option) in **NSObjCRuntime.h**.
- A new header and related functionality, **NSTask.h**, was added.
- See also changes noted in the section Notes Specific to OPENSTEP 4.0 PR2 Developers. All changes listed apply to OPENSTEP 4.0 PR1 developers.

Notes Specific to OPENSTEP 4.0 PR2 Developers

Items in the tables below (in the section titled *Tables Showing Foundation API Changes*) marked with "(40PR2)" are changes made to Foundation between OPENSTEP 4.0 PR2 and OPENSTEP 4.0.

Some additional changes are not contained in the tables or are of particular note. This list of changes applies to both OPENSTEP 4.0 PR1 and PR2 developers:

- The size of NSPort and NSRunLoop instances has changed. The NSPort class has become abstract and no longer has any

instance variables. Applications, tools, frameworks, bundles, or libraries with subclasses of these two classes must be recompiled.

- The header file **NSPosixFileDescriptor.h** has been removed from Foundation. This header was NeXT-specific. The header **NSFileHandle.h** provides portable replacement functionality.
- A few new headers have been added. The API in these headers is not included in the tables below:

NSAttributedString.h	NSFormatter.h
NSDateFormatter.h	NSNumberFormatter.h
NSEnumerator.h	NSProtocolChecker.h
NSFileHandle.h	

- Many declarations and definitions were removed from the header **NSUtilities.h** and moved elsewhere, but all the previous material is still defined by importing **NSUtilities.h**. The headers to which things were moved are **NSEnumerator.h**, **NSObjCRuntime.h**, **NSDictionary.h**, and **NSString.h**.
- The header **NSObjCRuntime.h** now imports the headers **objc/zone.h** and **stdarg.h**.
- The type of the parameter to NSTask's -**setStandardInput:**, **-setStandardOutput:**, and **-setStandardError:** methods has changed from **NSPosixFileDescriptor *** to **id**.
- The declarations of the NSAutoreleasePool debugging methods and aids have moved from **NSAutoreleasePool.h** to **NSDebug.h**. Some of the methods are destined for obsolescence, and some new methods have been added.
- These methods are no longer declared on Windows™ platforms:

CLASS	HEADER	METHOD
NSDictionary	NSFileManager.h	-fileSystemFileNumber
NSDictionary	NSFileManager.h	-fileSystemNumber
NSFileManager	NSFileManager.h	-createSymbolicLinkAtPath:pathContent:

NSFileManager	NSFileManager.h	-pathContentOfSymbolicLinkAtPath:

Although they are still declared on Windows™, an application should not use the string constants **NSFileSystemNumber** or **NSFileSystemFileNumber**, either.

Notes Specific to PDO 4.0 PR1 Developers

Items in the tables below (in the section titled *Tables Showing Foundation API Changes*) marked with "(P40PR1)" are changes made to Foundation between PDO 4.0 PR1 and OPENSTEP 4.0 and PDO 4.0.

Some additional changes are not contained in the tables or are of particular note:

- The Foundation is now shipped as a shared library in PDO 4.0 for Solaris™. All applications, tools, frameworks, bundles, libraries, etc. should be recompiled to take advantage of this.
- The size of NSPort and NSRunLoop instances has changed. The NSPort class has become abstract and no longer has any instance variables. Applications, tools, frameworks, bundles, or libraries with subclasses of these two classes must be recompiled.
- The header file **NSPosixFileDescriptor.h** has been removed from Foundation. This header was NeXT-specific. The header **NSFileHandle.h** provides portable replacement functionality.
- A few new headers have been added. The API in these headers is not included in the tables below:

NSAttributedString.h	NSFormatter.h
NSDateFormatter.h	NSNumberFormatter.h
NSFileHandle.h	NSProtocolChecker.h

- The type of the parameter to NSTask's -**setStandardInput:**, **-setStandardOutput:**, and **-setStandardError:** methods has

changed from **NSPosixFileDescriptor *** to **id**.

Tables Showing Foundation API Changes

There are four tables that list the API changes between previous versions of Foundation and the version of Foundation in OPENSTEP 4.0. The first table lists general changes to API elements (classes, macros, functions, constants, and so on) except methods, the second table lists the changes to the set of methods offered by the classes in Foundation, the third table lists changes to the return values of methods (for methods that didn't change name, which are listed in the second table), and the fourth table lists changes in the location of declarations. Other miscellaneous changes that weren't described well in table format are given in the sections above.

Changes are noted from the perspective of a developer who used a previous release looking to port their Foundation-based code to OPENSTEP 4.0 (and not through any intermediate releases). For instance, changes that are marked as having been made between OPENSTEP 4.0 PR1 and PR2 do not necessarily apply to NEXTSTEP 3.3 developers (for example, an API might have been added for OPENSTEP 4.0 PR1, but removed for OPENSTEP 4.0 PR2).

The rows in the tables are marked with various symbols in the rightmost column. One set of symbols describes between which releases and OPENSTEP 4.0 the change was made:

SYMBOL	MEANING
(NS33)	Change occurred between NEXTSTEP 3.3 and OPENSTEP 4.0
(EOF11)	Change occurred between EOF 1.1 and OPENSTEP 4.0
(40PR1)	Change occurred between OPENSTEP 4.0 PR1 and OPENSTEP 4.0
(40PR2)	Change occurred between OPENSTEP 4.0 PR2 and OPENSTEP 4.0

The other set of symbols indicate how the API changed. Both NeXT's implementation of Foundation, and the OpenStep

specification (since version 1.0 was published) have changed. These symbols attempt to describe what has happened:

SYMBOL	MEANING
(OpenStep,NeXT)	Generically, a change that did not add or remove API, but affects OpenStep and NeXT's implementation
(OpenStep)	Generically, a change that did not add or remove API and affects OpenStep, but not NeXT's implementation
(NeXT)	Generically, a change that did not add or remove API and affects NeXT's implementation, but not OpenStep
(+OpenStep,+NeXT)	Was in neither OpenStep nor NeXT's implementation, now in both
(+OpenStep,NeXT)	Was in NeXT's implementation only, now also in OpenStep
(+OpenStep,-NeXT)	Was in NeXT's implementation only, now in OpenStep only
(OpenStep,+NeXT)	Was in OpenStep only, now also in NeXT's implementation
(OpenStep,-NeXT)	Was in both, now in OpenStep only
(-OpenStep,+NeXT)	Was in OpenStep only, now in NeXT's implementation only
(-OpenStep,NeXT)	Was in both, now in NeXT's implementation only
(-OpenStep,-NeXT)	Was in both, now in neither
(+OpenStep)	Was in neither OpenStep nor NeXT's implementation, now in OpenStep only
(-OpenStep)	Was in OpenStep only, now in neither
(+NeXT)	Was in neither OpenStep nor NeXT's implementation, now in NeXT's implementation only
(-NeXT)	Was in NeXT's implementation only, now in neither

Occasionally, to the right of the rightmost column is a note for removed API which names the replacement.

GENERAL CHANGES IN PUBLIC API SYMBOLS

SYMBOL	TYPE	HEADER

ABS	macro	NSObjCRuntime.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
MAX	macro	NSObjCRuntime.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
MIN	macro	NSObjCRuntime.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSAccount	class	NSAccount.h	(-NeXT) (40PR1) (NS33)	
NSAdobeStandardCyrillicStringEncoding	constant	NSString.h	(-NeXT) (40PR2) (40PR1)	Use NSWindowsCP1251StringEncoding instead
NSAllocateMemoryPages	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArchiverArchiveInconsistency	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSArchiverClassError	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSArchiverDescriptorError	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSArchiverWriteReferenceError	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSArgumentInfo	typedef	NSMethodSignature.h		(-OpenStep,-NeXT) (40PR1)
NSBTreeBlock	class	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSBTreeComparator	typedef	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSBTreeCursor	class	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSBTreeStoreKeyTooLargeException	exception	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSBecomingMultiThreaded	notification	NSThread.h	(-OpenStep,-NeXT) (40PR1)	Use NSWillBecomeMultiThreadedNotification instead
NSBundleDidLoadNotification	notification	NSBundle.h	(+NeXT) (40PR1) (EOF11) (NS33)	
NSBundleDidLoadNotification	notification	NSBundle.h	(+OpenStep,NeXT) (40PR2)	
NSBundleLoaded	notification	NSBundle.h	(-NeXT) (40PR1)	Use NSBundleDidLoadNotification instead
NSByteStore	class	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSByteStoreDamagedException	exception	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSByteStoreFile	class	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSByteStoreLockedException	exception	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSByteStoreVersionException	exception	NSByteStore.h	(-OpenStep,-NeXT) (40PR1)	
NSCharacterConversionException	exception	NSString.h	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSCodingException	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSComparisonResult	typedef	NSObjCRuntime.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSConnectionDeath	notification	NSConnection.h	(-OpenStep,-NeXT) (40PR1)	Use NSConnectionDidDieNotification instead
NSConnectionDidInitializeNotification	notification	NSConnection.h	(+NeXT) (P40PR1) (40PR2) (40PR1)	
NSCopyMemoryPages	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSCurrencyString	default name	NSUserDefaults.h	(-OpenStep,NeXT) (40PR2)	Use NSCurrencySymbol instead
NSCurrencyString	default name	NSUserDefaults.h	(OpenStep,-NeXT) (40PR1)	Use NSCurrencySymbol instead
NSDateMissingTimeZone	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSDeallocateMemoryPages	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	

NSDecimalNumberBehavior	class	NSDecimalNumber.h	(-NeXT) (P40PR1) (40PR2) (40PR1)	
NSDecimalNumberBehavior	protocol	NSDecimalNumber.h	(-NeXT) (P40PR1) (40PR2) (40PR1)	
NSDecimalNumberBehaviors	protocol	NSDecimalNumber.h	(+NeXT) (P40PR1) (40PR2) (40PR1)	
NSDecimalNumberHandler	class	NSDecimalNumber.h	(+NeXT) (P40PR1) (40PR2) (40PR1)	
NSDistantObjectRequest	class	NSConnection.h	(+NeXT) (40PR1)	
NSEUCStringEncoding	constant	NSString.h	(-NeXT) (EOF11) (NS33)	Use NSJapaneseEUCStringEncoding instead
NSEqualRanges	function	NSRange.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSExceptionBase	constant	NSException.h	(-NeXT) (EOF11) (NS33)	
NSExtraRefCount	function	NSObject.h	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSFileGroupOwnerAccountName	dict. key	NSFileManager.h	(+NeXT) (40PR1)	
NSFileGroupOwnerAccountNumber	dict. key	NSFileManager.h	(-NeXT) (40PR1)	Use NSFileGroupOwnerAccountName instead
NSFileOwnerAccountName	dict. key	NSFileManager.h	(+NeXT) (40PR1)	
NSFileOwnerAccountNumber	dict. key	NSFileManager.h	(-NeXT) (40PR1)	Use NSFileOwnerAccountName instead
NSFullUserName	function	NSPathUtilities.h	(+NeXT) (40PR1) (EOF11) (NS33)	
NSGetSizeAndAlignment	function	NSObjCRuntime.h	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSGroupAccount	class	NSAccount.h	(-NeXT) (40PR1)	
NSHPUXOperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	
NSHashStringLength	constant	NSString.h	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33)	
NSISO2022JPStringEncoding	constant	NSString.h	(+NeXT) (P40PR1) (40PR2) (40PR1) (EOF11) (NS33)	
NSISOLatin2StringEncoding	constant	NSString.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSInconsistentArchiveException	exception	NSArchiver.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSInternationalCurrencyString	default name	NSUserDefaults.h	(OpenStep,+NeXT) (P40PR1) (40PR2)	
NSInternationalCurrencySymbol	default name	NSUserDefaults.h	(-NeXT) (P40PR1) (40PR2)	Use NSInternationalCurrencyString instead
NSIntersectsRect	function	NSGeometry.h	(+OpenStep,+NeXT) (40PR1)	
NSInvalidReceivePort	exception	NSException.h	(-NeXT) (40PR1)	Use NSInvalidReceivePortException instead
NSInvalidReceivePortException	exception	NSException.h	(+NeXT) (EOF11) (NS33)	
NSInvalidSendPort	exception	NSException.h	(-NeXT) (40PR1)	Use NSInvalidSendPortException instead
NSInvalidSendPortException	exception	NSException.h	(+NeXT) (EOF11) (NS33)	
NSJapaneseEUCStringEncoding	constant	NSString.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSLastException	constant	NSException.h	(-NeXT) (EOF11) (NS33)	
NSLoadedClasses	constant	NSBundle.h	(+OpenStep,+NeXT) (P40PR1) (40PR2) (40PR1) (EOF11) (NS33)	
NSLocalizedString	macro	NSBundle.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSLocalizedStringFromTable	macro	NSBundle.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSLocalizedStringFromTableInBundle	macro	NSBundle.h	(OpenStep,+NeXT) (EOF11) (NS33)	

NSLogPageSize	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSMACHOperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	
NSMakeRange	function	NSRange.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSMallocException	exception	NSException.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSNonRetainedObjectMapValueCallBacks	struct	NSMapTable.h	(+OpenStep,+NeXT) (40PR1)	
NSNotFound	constant	NSObjCRuntime.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSNotInArray	constant	NSArray.h	(-NeXT) (EOF11) (NS33)	
NSOSF1OperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	
NSObjCInadequateRuntimeInformation	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSOldStyleException	exception	NSException.h	(+NeXT) (EOF11) (NS33)	
NSOpenStepRootDirectory	function	NSPathUtilities.h	(+NeXT) (40PR1) (EOF11) (NS33)	
NSOpenStepUnicodeReservedBase	constant	NSCharacterSet.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSOwnedObjectIdentityHashCallBacks	struct	NSHashTable.h	(+OpenStep,+NeXT) (40PR1)	
NSOwnershipTransferException	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSPPLBecameDirty	notification	NSPPL.h	(-NeXT) (40PR1)	Use NSPPLDidBecomeDirtyNotification instead
NSPPLSaved	notification	NSPPL.h	(-NeXT) (40PR1)	Use NSPPLDidSaveNotification instead
NSPageSize	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSParseErrorException	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSPointFromString	function	NSGeometry.h	(+OpenStep,+NeXT) (40PR1)	
NSPortInvalidation	notification	NSPort.h	(-NeXT) (40PR1)	Use NSPortDidBecomeInvalidNotification instead
NSPortNameRegistrationError	exception	NSException.h	(-NeXT) (EOF11) (NS33)	
NSPortReceiveError	exception	NSException.h	(-NeXT) (40PR1)	Use NSPortReceiveException instead
NSPortReceiveException	exception	NSException.h	(+NeXT) (EOF11) (NS33)	
NSPortSendError	exception	NSException.h	(-NeXT) (40PR1)	Use NSPortSendException instead
NSPortSendException	exception	NSException.h	(+NeXT) (EOF11) (NS33)	
NSRangeFromString	function	NSRange.h	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSRealMemoryAvailable	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSRectFromString	function	NSGeometry.h	(+OpenStep,+NeXT) (40PR1)	
NSRoundDownToMultipleOfPageSize	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSRoundUpToMultipleOfPageSize	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSShiftJISStringEncoding	constant	NSString.h	(+NeXT) (NS33)	
NSShortDateFormatString	default name	NSUserDefaults.h	(+NeXT) (40PR1)	
NSSizeFromString	function	NSGeometry.h	(+OpenStep,+NeXT) (40PR1)	
NSSolarisOperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	

NSStandardApplicationPaths	function	NSPathUtilities.h	(+NeXT) (40PR1) (EOF11) (NS33)	
NSStandardLibraryPaths	function	NSPathUtilities.h	(+NeXT) (40PR1) (EOF11) (NS33)	
NSSunOSOperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	
NSTemporaryDirectory	function	NSPathUtilities.h	(+NeXT) (40PR1) (EOF11) (NS33)	
NSThreadExiting	notification	NSThread.h	(-OpenStep,-NeXT) (40PR1) (EOF11)	Use NSThreadWillExitNotification instead
NSThreadWillExitNotification	notification	NSThread.h	(+OpenStep,+NeXT) (40PR1) (EOF11)	
NSTimeIntervalSince1970	constant	NSDate.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSUTF8StringEncoding	constant	NSString.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSUTFStringEncoding	constant	NSString.h	(-NeXT) (EOF11) (NS33)	Use NSUTF8StringEncoding instead
NSUserAccount	class	NSAccount.h	(-NeXT) (40PR1)	
NSUserDefaultsChanged	notification	NSUserDefaults.h	(-OpenStep,-NeXT) (40PR1)	Use NSUserDefaultsDidChangeNotification instead
NSUserDefaultsDidChangeNotification	notification	NSUserDefaults.h	(+OpenStep,+NeXT) (40PR1)	
NSWillBecomeMultiThreadedNotification	notification	NSThread.h	(+OpenStep,+NeXT) (40PR1) (EOF11)	
NSWinLatin1StringEncoding	constant	NSString.h	(-NeXT) (40PR2) (40PR1)	Use NSWindowsCP1252StringEncoding instead
NSWindows95OperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	
NSWindowsCP1250StringEncoding	constant	NSString.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSWindowsCP1251StringEncoding	constant	NSString.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSWindowsCP1252StringEncoding	constant	NSString.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSWindowsCP1253StringEncoding	constant	NSString.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSWindowsCP1254StringEncoding	constant	NSString.h	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSWindowsNTOperatingSystem	constant	NSProcessInfo.h	(+NeXT) (40PR1)	
NSZoneFromPointer	function	NSZone.h	(OpenStep,+NeXT) (EOF11) (NS33)	
NSZoneFromPtr	function	NSZone.h	(-NeXT) (EOF11)	Use NSZoneFromPointer instead
NS_VALRETURN	macro	NSException.h	(-OpenStep,-NeXT) (EOF11) (NS33)	Use NS_VALUERETURN instead
NS_VALUERETURN	macro	NSException.h	(+OpenStep,+NeXT) (EOF11) (NS33)	
NXReadNSObjectFromCoder	function	NSCompatibility.h	(+NeXT) (P40PR1) (40PR2) (40PR1)	
Nil	constant	NSObjCRuntime.h	(+NeXT) (EOF11) (NS33)	
exception	variable	NSException.h	(-NeXT) (EOF11) (NS33)	Use localException instead for exception object in handler region
localException	variable	NSException.h	(OpenStep,+NeXT) (EOF11) (NS33)	
obsolete	macro	NSObject.h	(-NeXT) (EOF11) (NS33)	

GENERAL CHANGES IN PUBLIC API METHODS

CLASS/PROTOCOL	HEADER	METHOD

NSArchiver	NSArchiver.h	+classNameEncodedForTrueClassName:	(-NeXT) (EOF11) (NS33)	Use -classNameEncodedForTrueClassName: instead
NSArchiver	NSArchiver.h	-classNameEncodedForTrueClassName:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArchiver	NSArchiver.h	-encodeClassName:intoClassName:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArchiver	NSArchiver.h	-replaceObject:withObject:	(+NeXT) (EOF11) (NS33)	
NSArray	NSArray.h	+arrayWithArray:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	+arrayWithContentsOfFile:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	+arrayWithObjects:count:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-arrayByAddingObject:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArray	NSArray.h	-arrayByAddingObjectsFromArray:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArray	NSArray.h	-descriptionWithIndent:	(-NeXT) (EOF11) (NS33)	Use -descriptionWithLocale:indent: instead
NSArray	NSArray.h	-descriptionWithLocale:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArray	NSArray.h	-descriptionWithLocale:indent:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSArray	NSArray.h	-getObjects:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-getObjects:range:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-indexOfObject:inRange:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-indexOfObjectIdenticalTo:inRange:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-initWithContentsOfFile:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-sortedArrayHint	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-sortedArrayUsingFunction:context:hint:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSArray.h	-writeToFile:atomically:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSArray	NSPathUtilities.h	-pathsMatchingExtensions:	(+NeXT) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	+pathForResource:ofType:inDirectory:	(+OpenStep,+NeXT) (P40PR1) (40PR2) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	+pathForResource:ofType:inDirectory:withVersion:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	+setSystemLanguages:	(-NeXT) (EOF11) (NS33)	
NSBundle	NSBundle.h	+stripAfterLoading:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	-bundleVersion	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	-infoDictionary	(+NeXT) (EOF11) (NS33)	
NSBundle	NSBundle.h	-localizedStringForKey:value:comment:	(-NeXT) (EOF11) (NS33)	Use -localizedStringForKey:value:table: instead
NSBundle	NSBundle.h	-localizedStringForKey:value:comment:table:	(-NeXT) (EOF11) (NS33)	Use -localizedStringForKey:value:table: instead
NSBundle	NSBundle.h	-localizedStringForKey:value:table:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSBundle	NSBundle.h	-pathForResource:ofType:inDirectory:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	-pathsForResourcesOfType:inDirectory:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSBundle	NSBundle.h	-resourcePath;	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	

NSBundle	NSBundle.h	-setBundleVersion:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33)
NSCalendarDate	NSDate.h	+calendarDate	(OpenStep,+NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	+dateWithString:calendarFormat:locale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-addYear:month:day:hour:minute:second:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33) Use -dateByAddingYears:months:days:hours:minutes:seconds: instead
NSCalendarDate	NSDate.h	-dateByAddingYears:months:days:hours:minutes:seconds:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-dayOfCommonEra	(OpenStep,+NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-descriptionWithCalendarFormat:locale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-descriptionWithCalendarFormat:timeZone:	(-NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-descriptionWithLocale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-initWithString:calendarFormat:locale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	-years:months:days:hours:minutes:seconds:sinceDate:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSCharacterSet	NSCharacterSet.h	+characterSetWithContentsOfFile:	(+NeXT) (EOF11) (NS33)
NSCharacterSet	NSCharacterSet.h	+punctuationCharacterSet	(+NeXT) (EOF11) (NS33)
NSCharacterSet	NSCharacterSet.h	+punctuationCharacterSet	(+OpenStep,NeXT) (40PR1)
NSCoder	NSCoder.h	-decodeBytesWithReturnedLength:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSCoder	NSCoder.h	-encodeByrefObject:	(+NeXT) (40PR1) (EOF11) (NS33)
NSCoder	NSCoder.h	-encodeBytes:length:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSConnection	NSConnection.h	+currentConversation	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-addRequestMode:	(+OpenStep,+NeXT) (40PR1)
NSConnection	NSConnection.h	-addRunLoop:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-connection:shouldMakeNewConnection:	(+NeXT) (40PR1)
NSConnection	NSConnection.h	-createConversationForConnection:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-enableMultipleThreads	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-invalidate	(+OpenStep,+NeXT) (40PR1)
NSConnection	NSConnection.h	-localObjects	(+NeXT) (P40PR1) (40PR2)
NSConnection	NSConnection.h	-multipleThreadsEnabled	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-remoteObjects	(+NeXT) (P40PR1) (40PR2)
NSConnection	NSConnection.h	-removeRequestMode:	(+OpenStep,+NeXT) (40PR1)
NSConnection	NSConnection.h	-removeRunLoop:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-requestMode:	(-OpenStep,-NeXT) (40PR1)
NSConnection	NSConnection.h	-requestModes	(+OpenStep,+NeXT) (40PR1)
NSConnection	NSConnection.h	-runInNewThread	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSConnection	NSConnection.h	-setRequestMode::	(-OpenStep,-NeXT) (40PR1) Use -addRequestMode: instead

NSCopyingProtocol	NSObject.h	-copy	(-NeXT) (EOF11) (NS33)
NSData	NSData.h	+dataWithData:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+dateWithNaturalLanguageString:	(+NeXT) (EOF11) (NS33)
NSDate	NSDate.h	+dateWithNaturalLanguageString:locale:	(+NeXT) (EOF11) (NS33)
NSDate	NSDate.h	+dateWithString:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+dateWithTimeIntervalSince1970:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDate	NSDate.h	-descriptionWithCalendarFormat:timeZone:	(-NeXT) (EOF11) (NS33)
NSDate	NSDate.h	-descriptionWithCalendarFormat:timeZone:locale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDate	NSDate.h	-descriptionWithLocale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDate	NSDate.h	-isEqualToDate:	(+OpenStep,+NeXT) (40PR1)
NSDate	NSDate.h	-timeIntervalSince1970	(OpenStep,+NeXT) (EOF11) (NS33)
NSDecimalNumber	NSDecimalNumber.h	+decimalNumberBehaviorWithRoundingMode:scale:raiseOnExactness:raiseOnOverflow:raiseOnUnderflow:raiseOnDivideByZero:	(-NeXT) (P40PR1) (40PR2) (40PR1) Use +decimalNumberHandlerWithRoundingMode:... instead
NSDecimalNumber	NSDecimalNumber.h	+decimalNumberHandlerWithRoundingMode:scale:raiseOnExactness:raiseOnOverflow:raiseOnUnderflow:raiseOnDivideByZero:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-decimalNumberBySubstracting:	(-NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-decimalNumberBySubstracting:withBehavior:	(-NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-decimalNumberBySubtracting:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-decimalNumberBySubtracting:withBehavior:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-maximumDecimalNumber	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-minimumDecimalNumber	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumberBehavior	NSDecimalNumber.h	+defaultDecimalNumberBehavior	(-NeXT) (P40PR1) (40PR2) (40PR1)
NSDecimalNumberHandler	NSDecimalNumber.h	+defaultDecimalNumberHandler	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSDictionary	NSDictionary.h	+dictionaryWithContentsOfFile:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDictionary	NSDictionary.h	+dictionaryWithDictionary:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSDictionary	NSDictionary.h	+dictionaryWithObject:forKey:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSDictionary	NSDictionary.h	+dictionaryWithObjectsAndKeys:...	(OpenStep,+NeXT) (EOF11) (NS33)
NSDictionary	NSDictionary.h	-descriptionWithIndent:	(-NeXT) (EOF11) (NS33) Use -descriptionWithLocale:indent: instead
NSDictionary	NSDictionary.h	-descriptionWithLocale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDictionary	NSDictionary.h	-descriptionWithLocale:indent:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDictionary	NSDictionary.h	-initWithObjects:forKeys:	(OpenStep,+NeXT) (EOF11) (NS33)
NSDictionary	NSDictionary.h	-initWithObjectsAndKeys:...	(OpenStep,+NeXT) (EOF11) (NS33)
NSDictionary	NSDictionary.h	-keysSortedByValueUsingSelector:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSDictionary	NSDictionary.h	-objectsForKeys:notFoundMarker:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)

NSDictionary	NSFileManager.h	-fileGroupOwnerAccountName	(+NeXT) (40PR1)	
NSDictionary	NSFileManager.h	-fileGroupOwnerAccountNumber	(-NeXT) (40PR1)	Use -fileGroupOwnerAccountName instead
NSDictionary	NSFileManager.h	-fileOwnerAccountName	(+NeXT) (40PR1)	
NSDictionary	NSFileManager.h	-fileOwnerAccountNumber	(-NeXT) (40PR1)	Use -fileOwnerAccountName instead
NSDistantObjectRequest	NSConnection.h	-conversation	(+NeXT) (P40PR1) (40PR2)	Note class was added after OPENSTEP 4.0 PR1
NSEnumerator	NSEnumerator.h	-allObjects	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h
NSException	NSException.h	-exceptionName	(-NeXT) (EOF11) (NS33)	
NSException	NSException.h	-exceptionReason	(-NeXT) (EOF11) (NS33)	
NSException	NSException.h	-exceptionUserInfo	(-NeXT) (EOF11) (NS33)	
NSException	NSException.h	-name	(OpenStep,+NeXT) (EOF11) (NS33)	
NSException	NSException.h	-reason	(OpenStep,+NeXT) (EOF11) (NS33)	
NSException	NSException.h	-userInfo	(OpenStep,+NeXT) (EOF11) (NS33)	
NSMethodSignature	NSMethodSignature.h		+signatureWithObjCTypes:	(-OpenStep,-NeXT) (40PR1)
NSMethodSignature	NSMethodSignature.h		-argumentInfoAtIndex:	(-OpenStep,-NeXT) (40PR1) Use -getArgumentTypeAtIndex: instead
NSMethodSignature	NSMethodSignature.h		-getArgumentTypeAtIndex:	(+OpenStep,+NeXT) (40PR1)
NSMutableArray	NSArray.h	-removeObject:inRange:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSMutableArray	NSArray.h	-removeObjectIdenticalTo:inRange:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSMutableArray	NSArray.h	-removeObjectsInRange:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSMutableArray	NSArray.h	-replaceObjectsInRange:withObjectsFromArray:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSMutableArray	NSArray.h	-replaceObjectsInRange:withObjectsFromArray:range:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)	
NSMutableArray	NSArray.h	-setArray:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSMutableArray	NSArray.h	-sortUsingSelector:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSMutableCopyingProtocol	NSObject.h	-mutableCopy	(-NeXT) (EOF11) (NS33)	
NSMutableData	NSData.h	-setData:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)	
NSMutableDictionary	NSDictionary.h	-setDictionary:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSMutableSet	NSSet.h	-setSet:	(+NeXT) (40PR2) (40PR1)	
NSNotification	NSNotification.h	+notificationWithName:object:userInfo:	(OpenStep,+NeXT) (EOF11) (NS33)	
NSNotification	NSNotification.h	-name	(OpenStep,+NeXT) (EOF11) (NS33)	
NSNotification	NSNotification.h	-notificationName	(-NeXT) (EOF11) (NS33)	
NSNotification	NSNotification.h	-notificationObject	(-NeXT) (EOF11) (NS33)	
NSNotification	NSNotification.h	-object	(OpenStep,+NeXT) (EOF11) (NS33)	
NSNotification	NSNotification.h	-userInfo	(OpenStep,+NeXT) (EOF11) (NS33)	

NSNotificationCenter	NSNotification.h	-addObserver:selector:name:object:	(OpenStep,+NeXT) (EOF11) (NS33)
NSNotificationCenter	NSNotification.h	-addObserver:selector:notificationName:object:	(-NeXT) (EOF11) (NS33)
NSNotificationCenter	NSNotification.h	-postNotificationName:object:userInfo:	(OpenStep,+NeXT) (EOF11) (NS33)
NSNotificationCenter	NSNotification.h	-removeObserver:name:object:	(OpenStep,+NeXT) (EOF11) (NS33)
NSNotificationCenter	NSNotification.h	-removeObserver:notificationName:object:	(-NeXT) (EOF11) (NS33)
NSNumber	NSValue.h	-descriptionWithLocale:	(+NeXT) (EOF11) (NS33)
NSNumber	NSValue.h	-descriptionWithLocale:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithBool:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithBool:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithChar:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithChar:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithDouble:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithDouble:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithFloat:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithFloat:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithInt:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithInt:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithLong:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithLong:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithLongLong:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithLongLong:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithShort:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithShort:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithUnsignedChar:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithUnsignedChar:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithUnsignedInt:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithUnsignedInt:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithUnsignedLong:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithUnsignedLong:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithUnsignedLongLong:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithUnsignedLongLong:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-initWithUnsignedShort:	(+NeXT) (NS33)
NSNumber	NSValue.h	-initWithUnsignedShort:	(+OpenStep,NeXT) (40PR1)
NSNumber	NSValue.h	-isEqualToNumber:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)

NSObject	NSConnection.h	-connection:handleRequest:	(+NeXT) (40PR1)
NSObject	NSObject.h	+copyWithZone:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSObject	NSObject.h	+load	(+NeXT) (40PR1)
NSObject	NSObject.h	+load	(+OpenStep,NeXT) (P40PR1) (40PR2)
NSObject	NSObject.h	+mutableCopyWithZone:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSObject	NSObject.h	-copy	(OpenStep,+NeXT) (EOF11) (NS33)
NSObject	NSObject.h	-description	(-NeXT) (EOF11) (NS33)
NSObject	NSObject.h	-mutableCopy	(OpenStep,+NeXT) (EOF11) (NS33)
NSObject	NSRunLoop.h	-performSelector:object:afterDelay:	(-OpenStep,-NeXT) (40PR1)
NSObject	NSRunLoop.h	-performSelector:withObject:afterDelay:	(+OpenStep,+NeXT) (40PR1)
NSObject	NSRunLoop.h	-performSelector:withObject:afterDelay:inModes:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSObjectProtocol	NSObject.h	-description	(OpenStep,+NeXT) (EOF11) (NS33)
NSObjectProtocol	NSObject.h	-perform:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33) Use -performSelector: instead
NSObjectProtocol	NSObject.h	-perform:withObject:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33) Use -performSelector:withObject: instead
NSObjectProtocol	NSObject.h	-perform:withObject:withObject:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33) Use -performSelector:withObject:withObject: instead
NSObjectProtocol	NSObject.h	-performSelector:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSObjectProtocol	NSObject.h	-performSelector:withObject:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSObjectProtocol	NSObject.h	-performSelector:withObject:withObject:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSPort	NSPort.h	-addConnection:toRunLoop:forMode:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPort	NSPort.h	-removeConnection:fromRunLoop:forMode:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPort	NSPort.h	-reservedSpaceLength	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPort	NSPort.h	-sendBeforeDate:components:from:reserved:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPortCoder	NSPortCoder.h	+portCoderWithReceivePort:sendPort:components:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPortCoder	NSPortCoder.h	-dispatch	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPortCoder	NSPortCoder.h	-isByref	(+NeXT) (40PR1)
NSPortNameServer	NSPortNameServer.h	-portForName:host:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSPortNameServer	NSPortNameServer.h	-registerPort:name:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSProcessInfo	NSProcessInfo.h	-operatingSystem	(+NeXT) (40PR1)
NSProxy	NSProxy.h	+load	(+NeXT) (40PR1)
NSProxy	NSProxy.h	+load	(+OpenStep,NeXT) (P40PR1) (40PR2)
NSProxy	NSProxy.h	+respondsToSelector:	(+NeXT) (40PR1)
NSRunLoop	NSRunLoop.h	-addPosixFileDescriptor:forMode:	(-NeXT) (P40PR1) (40PR2) (40PR1)

NSRunLoop	NSRunLoop.h	-addRunLoopActivityMonitor:	(-NeXT) (40PR1)
NSRunLoop	NSRunLoop.h	-cancelPerformSelector:target:argument:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSRunLoop	NSRunLoop.h	-performSelector:target:argument:order:modes:	(+NeXT) (P40PR1) (40PR2) (40PR1)
NSRunLoop	NSRunLoop.h	-removePosixFileDescriptor:forMode:	(-NeXT) (P40PR1) (40PR2) (40PR1)
NSScanner	NSScanner.h	+localizedScannerWithString:	(OpenStep,+NeXT) (EOF11) (NS33)
NSScanner	NSScanner.h	-locale	(OpenStep,+NeXT) (EOF11) (NS33)
NSScanner	NSScanner.h	-scanHexInt:	(+NeXT) (EOF11) (NS33)
NSScanner	NSScanner.h	-scanHexInt:	(+OpenStep,NeXT) (40PR1) (EOF11) (NS33)
NSScanner	NSScanner.h	-setLocale:	(OpenStep,+NeXT) (EOF11) (NS33)
NSSet	NSSet.h	+setWithObjects:count:	(+NeXT) (40PR2) (40PR1)
NSSet	NSSet.h	+setWithSet:	(+NeXT) (40PR2) (40PR1)
NSString	NSPathUtilities.h	+pathWithComponents:	(+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSPathUtilities.h	-fileSystemRepresentation	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSPathUtilities.h	-getFileSystemRepresentation:maxLength:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSPathUtilities.h	-isAbsolutePath	(+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSPathUtilities.h	-pathComponents	(+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSPathUtilities.h	-stringsByAppendingPaths:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSString.h	+availableStringEncodings	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	+localizedNameOfStringEncoding:	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	+localizedStringWithFormat:...	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	+string	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSString	NSString.h	+stringWithContentsOfFile:	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	+stringWithString:	(+NeXT) (40PR2) (40PR1) (EOF11) (NS33)
NSString	NSString.h	-caseInsensitiveCompare:	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	-doubleValue	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	-getLineStart:end:contentsEnd:forRange:	(+NeXT) (P40PR1) (40PR2) (40PR1) (EOF11) (NS33)
NSString	NSString.h	-initWithFormat:locale:...	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	-initWithFormat:locale:arguments:	(OpenStep,+NeXT) (EOF11) (NS33)
NSString	NSString.h	-lineRangeForRange:	(+NeXT) (P40PR1) (40PR2) (40PR1) (EOF11) (NS33)
NSString	NSString.h	-lossyCString	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSString.h	-substringFromRange:	(-OpenStep,-NeXT) (40PR1) (EOF11) (NS33) Use -substringWithRange: instead
NSString	NSString.h	-substringWithRange:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
NSString	NSString.h	-writeToFile:atomically:	(OpenStep,+NeXT) (EOF11) (NS33)
NSThread	NSThread.h	+exit	(OpenStep,+NeXT) (EOF11)

NSThread	NSThread.h	+sleepUntilDate:	(OpenStep,+NeXT) (EOF11)
NSThread	NSThread.h	-exit	(-NeXT) (EOF11)
NSTimer	NSTimer.h	-isValid	(+OpenStep,+NeXT) (40PR1)
NSUnarchiver	NSArchiver.h	-classNameDecodedForArchiveClassName:	(OpenStep,+NeXT) (EOF11) (NS33)
NSUnarchiver	NSArchiver.h	-decodeClassName:asClassName:	(OpenStep,+NeXT) (EOF11) (NS33)
NSUnarchiver	NSArchiver.h	-replaceObject:withObject:	(+NeXT) (EOF11) (NS33)
NSUserDefaults	NSUserDefaults.h	-setSearchList:	(+OpenStep,+NeXT) (40PR1)
NSValue	NSValue.h	+valueWithBytes:objCType:	(+NeXT) (EOF11) (NS33)
NSValue	NSValue.h	+valueWithBytes:objCType:	(+OpenStep,NeXT) (40PR1)
NSValue	NSValue.h	-initWithBytes:objCType:	(+NeXT) (EOF11) (NS33)
NSValue	NSValue.h	-initWithBytes:objCType:	(+OpenStep,NeXT) (40PR1)
NSValue	NSValue.h	-isEqualToValue:	(+OpenStep,+NeXT) (40PR1) (EOF11) (NS33)
Object	NSCompatibility.h	+allocWithZone:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	+instanceMethodForSelector:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	+instancesRespondToSelector:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	+poseAsClass:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-autorelease	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-conformsToProtocol:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-copyWithZone:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-dealloc	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-doesNotRecognizeSelector:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-isKindOfClass:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-isMemberOfClass:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-methodForSelector:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-perform:withObject:	(-NeXT) (40PR1) (EOF11) (NS33) Use -performSelector:withObject: instead
Object	NSCompatibility.h	-perform:withObject:withObject:	(-NeXT) (40PR1) Use -performSelector:withObject:withObject: instead
Object	NSCompatibility.h	-performSelector:withObject:	(+NeXT) (40PR1)
Object	NSCompatibility.h	-performSelector:withObject:withObject:	(+NeXT) (40PR1)
Object	NSCompatibility.h	-release	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-respondsToSelector:	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-retain	(+NeXT) (EOF11) (NS33)
Object	NSCompatibility.h	-retainCount	(+NeXT) (EOF11) (NS33)

METHOD RETURN TYPE CHANGES

CLASS/PROTOCOL	HEADER	METHOD	OLD RET. TYPE	NEW RET. TYPE	
NSBundle	NSBundle.h	-classNamed:	(id)	(Class)	(NeXT) (EOF11) (NS33)
NSBundle	NSBundle.h	-principalClass	(id)	(Class)	(NeXT) (EOF11) (NS33)
NSCalendarDate	NSDate.h	+calendarDate	(NSCalendarDate *)	(id)	(NeXT) (40PR1)
NSCalendarDate	NSDate.h	+dateWithString:calendarFormat:	(NSCalendarDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSCalendarDate	NSDate.h	+dateWithString:calendarFormat:locale:	(NSCalendarDate *)	(id)	(NeXT) (40PR1)
NSCalendarDate	NSDate.h	+dateWithYear:month:day:hour:minute:second:timeZone:		(NSCalendarDate *)	(id) (NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+date	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+dateWithTimeIntervalSince1970:	(NSDate *)	(id)	(NeXT) (40PR1)
NSDate	NSDate.h	+dateWithTimeIntervalSinceNow:	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+dateWithTimeIntervalSinceReferenceDate:	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+distantFuture	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	+distantPast	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	-addTimeInterval:	(NSDate *)	(id)	(NeXT) (EOF11) (NS33)
NSDate	NSDate.h	-initWithTimeInterval:sinceDate:	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDate	NSDate.h	-initWithTimeIntervalSinceNow:	(NSDate *)	(id)	(NeXT) (40PR1) (EOF11) (NS33)
NSDecimalNumber	NSDecimalNumber.h	-initWithDecimal:	(NSDecimalNumber *)	(id)	(OpenStep,NeXT) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-initWithMantissa:exponent:isNegative:	(NSDecimalNumber *)	(id)	(OpenStep,NeXT) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-initWithString:	(NSDecimalNumber *)	(id)	(OpenStep,NeXT) (40PR2) (40PR1)
NSDecimalNumber	NSDecimalNumber.h	-initWithString:locale:	(NSDecimalNumber *)	(id)	(OpenStep,NeXT) (40PR2) (40PR1)
NSDictionary	NSFileManager.h	-filePosixPermissions	(unsigned int)	(unsigned long)	(NeXT) (40PR1)
NSDictionary	NSFileManager.h	-fileSize	(long long)	(unsigned long long)	(NeXT) (40PR1)
NSDictionary	NSFileManager.h	-fileSystemFileNumber	(unsigned int)	(unsigned long)	(NeXT) (40PR1)
NSDictionary	NSFileManager.h	-fileSystemNumber	(unsigned int)	(unsigned long)	(NeXT) (40PR1)
NSMethodSignature	NSMethodSignature.h		-methodReturnType	(char *)	(const char *) (OpenStep,NeXT) (40PR1)
NSMutableString	NSString.h	+stringWithCapacity:	(NSMutableString *)	(id)	(OpenStep,NeXT) (40PR1) (EOF11) (NS33)
NSNotification	NSNotification.h	+notificationWithName:object:	(NSNotification *)	(id)	(NeXT) (EOF11) (NS33)
NSObject	NSObject.h	+initialize	(id)	(void)	(NeXT) (EOF11) (NS33)
NSObject	NSObject.h	+load	(id)	(void)	(NeXT) (EOF11) (NS33)
NSObject	NSObject.h	+superclass	(id)	(Class)	(NeXT) (EOF11) (NS33)
NSObjectProtocol	NSObject.h	-superclass	(id)	(Class)	(NeXT) (EOF11) (NS33)

NSPPL	NSPPL.h	+pplWithPath:create:readOnly:	(NSPPL *)	(id)	(NeXT) (40PR1)
NSPPL	NSPPL.h	+pplWithPath:fromPPLData:readOnly:	(NSPPL *)	(id)	(NeXT) (40PR1)
NSString	NSString.h	+localizedStringWithFormat:...	(NSString *)	(id)	(OpenStep,NeXT) (40PR1)
NSString	NSString.h	+stringWithCString:	(NSString *)	(id)	(OpenStep,NeXT) (40PR1) (EOF11) (NS33)
NSString	NSString.h	+stringWithCString:length:	(NSString *)	(id)	(OpenStep,NeXT) (40PR1) (EOF11) (NS33)
NSString	NSString.h	+stringWithCharacters:length:	(NSString *)	(id)	(OpenStep,NeXT) (40PR1) (EOF11) (NS33)
NSString	NSString.h	+stringWithContentsOfFile:	(NSString *)	(id)	(OpenStep,NeXT) (40PR1)
NSString	NSString.h	+stringWithFormat:...	(NSString *)	(id)	(OpenStep,NeXT) (40PR1) (EOF11) (NS33)
NSTask	NSTask.h	-standardInput	(NSPosixFileDescriptor *)	(id)	(NeXT) (P40PR1) (40PR2)
NSTask	NSTask.h	-standardOutput	(NSPosixFileDescriptor *)	(id)	(NeXT) (P40PR1) (40PR2)
NSTask	NSTask.h	-standardError	(NSPosixFileDescriptor *)	(id)	(NeXT) (P40PR1) (40PR2)
NSTask	NSTask.h	+launchedTaskWithLaunchPath:arguments:	(id)	(NSTask *)	(NeXT) (P40PR1) (40PR2)
NSTimeZone	NSDate.h	+timeZoneWithAbbreviation:	(NSTimeZone *)	(NSTimeZoneDetail *)	(NeXT) (EOF11) (NS33)
NSUserDefaults	NSUserDefaults.h	-searchList	(NSMutableArray *)	(NSArray *)	(OpenStep,NeXT) (40PR1)
Object	NSCompatibility.h	+poseAsClass:	(id)	(void)	(NeXT) (NS33)

CHANGES TO API DECLARATION LOCATION

SYMBOL	OLD HEADER	NEW HEADER		
ABS	NSUtilities.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1)	
MAX	NSUtilities.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1)	
MIN	NSUtilities.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1)	
NSCharacterConversionException	NSException.h	NSString.h	(NeXT) (40PR1)	
NSComparisonResult	NSObject.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSObject.h
NSEnumerator class	NSUtilities.h	NSEnumerator.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h
NSLog()	NSUtilities.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h
NSLogv()	NSUtilities.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h
NSNotFound	NSObject.h	NSObjCRuntime.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSObject.h
NXReadNSObject()	NSArchiver.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
NXWriteNSObject()	NSArchiver.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[NSCoder -decodeNXObject:]	NSArchiver.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[NSCoder -encodeNXObject:]	NSArchiver.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[NSDictionary -descriptionInStringsFileFormat]	NSUtilities.h	NSDictionary.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h
[NSString -propertyListFromStringsFileFormat]	NSUtilities.h	NSString.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h

[NSString -propertyList]	NSUtilities.h	NSString.h	(NeXT) (40PR2) (40PR1) (EOF11) (NS33)	For OpenStep compatibility, import NSUtilities.h
[Object +allocWithZone:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object +instanceMethodForSelector:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object +instancesRespondToSelector:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object +poseAsClass:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -autorelease]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -conformsToProtocol:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -copyWithZone:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -dealloc]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -doesNotRecognizeSelector:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -isKindOfClass:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -isMemberOfClass:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -methodForSelector:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -release]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -respondsToSelector:]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -retainCount]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	
[Object -retain]	NSUtilities.h	NSCompatibility.h	(NeXT) (EOF11) (NS33)	

General Comments

Deep vs. Shallow Copies

In various places within the Foundation reference (primarily in the reference documentation for the collection classes), statements similar to the following are made in reference to ªdeep copiesº:

> ªdeepº means that all of the objects in the dictionary are copied, and all of the objects they contain

are copied, and so on.

At a broad level, the above explanation is true. Strictly speaking, however, it is not and never has been. Immutable objects do a **[self retain]** when told to **copyWithZone:**. For leaf objects like NSStrings, this is quite reasonable. For collections, though, it has the following effect: as soon as an immutable collection is reached in the deep copy traversal of some object heirarchy, the immutable collection simply retains itself. Thus any mutable objects *it* might contain *are not copied*.

Foundation Examples

There are several Foundation examples in **/NextDeveloper/Examples/Foundation**.

Various examples illustrating the use of Distributed OLE and the NeXT ORB can be found on NeXTanswers. NeXTanswers can be accessed through NeXT's Web page, at http://www.next.com.

I/O Functionality

Two new Foundation classes, NSFileHandle and NSPipe, provide objects that represent open files or communications channels and support basic I/O operations. They make it easy for developers to write code that is directly portable to other platforms. The interfaces for these classes are declared in **NSFileHandle.h**. These classes replace the NSPosixFileDescriptor and NSPosixPipeDescriptor classes made available in OPENSTEP 4.0 prereleases. The differences from the predecessor classes for the most part involve naming and a refinement of semantics.

Archiving

NSUnarchivers own the objects that they decode. When an NSUnarchiver is deallocated, so are the objects that it has decoded, unless they have been retained. The following code, for example, will probably result in a decoded object being used after being freed:

```
NSAutoreleasePool *pool = [[NSAutoreleasePool allocWithZone:NULL] init];
NSUnarchiver *unarchiver = [[NSUnarchiver allocWithZone:NULL] initForReadingWithData:myData];
id object = [unarchiver decodeObject];
[unarchiver release];
[pool release];
... use or return object here ...
```

`object` should be retained before the NSUnarchiver is released, and probably also autoreleased if `object` is returned from the function or method. An exception to this are the **-decodeValueOfObjCType:at:** and **-decodeValueOfObjCTypes:...** methods. Objects "returned" from these two methods, like those returned from **+allocWithZone:** and **-copyWithZone:**, are not autoreleased, and must be explicitly released.

User Defaults

Foundation contains API (NSUserDefaults) which replaces the old defaults database suite of functions. User defaults with 4.0 applications are stored differently and in a different place than 3.3 user defaults. There is a new command-line utility called **defaults** that allows you to manipulate the new-style defaults just as **dread**, **dwrite**, and **dremove** allowed you to manipulate old-style defaults.

On Windows NT™, defaults are stored in the Windows™ registry. There is a program shipped with Windows NT™, **REGEDT32**, for viewing and editing registry entries. NSUserDefaults stores its domains and their key-value pairs under the entry **HKEY_CURRENT_USER\Software\NeXT\UserDefaults**. The values are currently stored as strings in the property list format.

The command **/usr/bin/defconvert**, in OPENSTEP 4.0 for MACH, can be used to convert a user's 3.3 defaults database to 4.0-style defaults±simply run it from the command line with no parameters. This program converts all old-style defaults, even those of non-OpenStep applications, adding them to the user's new-style database. Since NEXSTEP 3.3 applications may not use the same default names after conversion to OPENSTEP 4.0, and 3.3 applications do not use the new defaults database, and increasing the size of the defaults database increases the amount of memory needed by each OpenStep application, converting an old defaults database to the new style usually provides little utility.

Errors in ASCII Property Lists

When a method like NSDictionary's +**dictionaryWithContentsOfFile:** is used to read and parse an ASCII property list from a file, parse errors and exceptions are suppressed, and **nil** is returned upon error. Sometimes a program wants to know about syntax errors in a property list, or wants finer-grained information about why a property list cannot be read and parsed. Another method, NSString's -**propertyList**, can be used for this purpose. The following example illustrates this:

```
- (id)readPropertyListFromFile:(NSString *)path mustBeOfType:(Class)targetClass logErrors:(BOOL)showErrors {
    NSString *string;
    id plist = nil;

    string = [[NSString allocWithZone:NULL] initWithContentsOfFile:path];
    if (showErrors && nil == string) {
        NSLog(@"%@: string could not be read from '%@'", NSStringFromSelector(_cmd), path);
    }
    NS_DURING
        plist = [string propertyList];
    NS_HANDLER
        if (showErrors) {
```

```
            NSLog(@"%@: received exception while parsing: %@", NSStringFromSelector(_cmd), localException);
        }
        plist = nil;
    NS_ENDHANDLER
    [string release];
    if (Nil != targetClass && ![plist isKindOfClass:targetClass]) {
        if (showErrors) {
            NSLog(@"%@: property list is not of desired type '%@'. It's an '%@'.",
                  NSStringFromSelector(_cmd), NSStringFromClass(targetClass), NSStringFromClass([plist class]));
        }
        [plist release];
        plist = nil;
    }
    return plist;
}
```

Grammars for Serialization and ASCII Property List Formats

Grammars and description of the serialization (NSSerializer) and ASCII property list representation formats are available in the appendix to the OpenStep specification.

Strings which contain non-alphanumeric characters must be double quoted in property lists to ensure the entire string is parsed as one string object. Parse failures will result, otherwise.

Retain Cycles and Invalidation

The retain/release strategy used in Foundation allows an incautious programmer to create reference cycles in an object graph

(that is, the graph of objects that retain one another in some manner). These cycles are self-sustaining and constitute a memory leak. The simplest example is to create a NSMutableArray instance and add it to itself. Since an array retains its objects, it retains itself independent of its normal usage. Retains are essentially distributed across the network via the Distributed Objects mechanism.

If retain cycles cannot be avoided by careful design, but are known *a priori*, the following technique can be used to recover the memory. For the objects in the cycle(s) whose retain counts are solely due to the cycle, all should be retained (perhaps in an array), each one told to release its retained objects, then all released. The action of releasing its retained objects would undoubtedly cause an object to become dysfunctional. Some objects already implement such a method by the name "-invalidate". It is a bug that neither NSObject nor the collections implement this method to facilitate cycle recovery.

More on Autoreleasing and Retaining

The following statements are false:

Returned objects are guaranteed to be valid for the scope of the current method.
Returned objects are guaranteed to be valid until the current autorelease pool is released.
Returned objects are guaranteed to be valid until the end of the current event loop.

The Foundation's retain count mechanism operates via **-retain** and **-release**. A code fragment such as:

```
id object = [collection returnObject];
```

creates a reference (in `object`) to the returned object, but does not increment the retain count of `object`. If you don't let the system know about your reference, by incrementing the retain count with **-retain**, the system can't ensure that your reference remains valid for any length of time. In practice, you can get away with such "weak references" most of the time. But the safest approach formalizes your reference to the object:

```
id object = [[collection returnObject] retain];
/* Do operations, some of which may be on 'object' */
...
/* Don't need object any longer */
[object release];
object = nil;
```

The -hash and -isEqual: Invariant

It is worthwhile to highlight an obscure requirement documented in NSObject class specification for the **-hash** method: The **-hash** and **-isEqual:** methods of a class must be defined so that if two objects are equal, their hash value is the same ([x isEqual:y] implies [x hash] == [y hash], but not the converse). This is easy to forget, but especially important to ensure when putting custom objects into some collections (such as NSSet and NSDictionary).

Forwarding Messages in OpenStep

Objects wishing to forward messages with the **-forwardInvocation:** method must also (re)implement the **-methodSignatureForSelector:** method. This is a result of the compiler not providing call stack descriptive information at each call site. Consequently, the runtime needs a source of information for how the stack frame is constructed. It needs this to build the invocation that will be supplied to the **-forwardInvocation:** method call. The **-methodSignatureForSelector:** method will need to be reimplemented to provide a method signature for selectors to which the object does not respond.

Fault objects in EOF which wish to forward messages should also override **+instanceMethodSignatureForSelector:**.

```
- (NSMethodSignature *)methodSignatureForSelector:(SEL)sel {
    NSMethodSignature *result = [super methodSignatureForSelector:sel];
```

```
    if (nil == result) {    // must not respond to the selector, no runtime information available
        result = [target methodSignatureForSelector:sel];
    return result;
}

- (void)forwardInvocation:(NSInvocation *)invocation {
    [invocation invokeWithTarget:target];  // let target implement methods that we do not
}
```

See the ForwardInvocation example in **/NextDeveloper/Examples/Foundation/ForwardInvocation**.

Debugging Aids

The *OpenStep Conversion Guide* describes several techniques for debugging Foundation and AppKit applications. The header file **NSDebug.h** also has some global flags, functions, and methods which may be useful. But be warned±although this header file is public, its contents are mostly unsupported, largely undocumented, and are subject to change without warning. Do not depend on API within this header for the operation of production programs.

Object Allocation Analysis

One of the useful flags defined in **NSDebug.h** is **NSKeepAllocationStatistics**, which allows you to analyze the object allocation activity in a program. The **AnalyzeAllocation** command line program can be used to analyze the output generated by running with this option enabled. You can also use the **ObjectAlloc** demo application which comes OPENSTEP for Mach and Windows to graph the object allocation activity in real time.

The +load Method

The **+load** message is sent to classes when they are added to the Objective C runtime. **+load** is usually received before **+initialize**. **+load** is not inherited by subclasses.

The order in which **+load** messages are sent to classes is unspecified (and specifically, a class's superclass is not guaranteed to have its **+load** method called before the class receives the message). Therefore, you should not use any subsystems or classes which are loaded at the same time, or more generally, which you do not know have received the **+load** message. If you do, you will probably cause execution of code which assumes that **+load** (and anything that *had* to be done in **+load**) has been previously executed.

There are two situations typically when classes are added to the runtime:

- when loaded dynamically from a bundle or framework by NSBundle
- at application launch

The restrictions on **+load** are perhaps most serious at launch time. The Foundation has classes and subsystems (as might any other library) which depend on **+load** being called before they are fully functional. For example, you should not use **NSZoneMalloc()** to allocate memory in **+load** methods which are statically linked into an application. You certainly should not create any objects.

Note that calling **+load** on a class isn't a solution (and is a generally bad idea), because that will cause **+initialize** to be sent first, which the class may not handle. Not to mention that doing so places into your code dependencies on which classes implement **+load** in a particular version of a library.

+load is not intended as a general initialization mechanism, and should only be used when absolutely necessary. Use **+initialize** for early initialization whenever possible.

Performance and NSNotificationCenter

NSNotificationCenter, although heavily optimized, by its nature can become a significant performance bottleneck if heavy use of notifications is made. The notification mechanism should be used judiciously.

NSAutoreleasePool's -addObject: Method

You should never use the NSAutoreleasePool instance method **-addObject:**. Use the **-autorelease** method, or **[NSAutoreleasePool addObject:]** to autorelease an object. Using the instance method incurs the risk of adding an object to a pool which is not the top autorelease pool, a semantically suspect operation, since the autorelease pool model is one of a (per-thread) stack of pools. OPENSTEP 4.0 does allow you to add objects to an autorelease pool that is not the top pool, but be warned that it is a much more expensive operation (and always will be) than adding an object to the top pool.

Frameworks as NSBundles

Frameworks are a specialized form of bundle. An application can statically link against a framework at compile time (like a library), or dynamically load a framework (like a bundle) at runtime. However, as a specialized type of bundle, there are some constraints placed on frameworks:

- A framework must have a **.framework** extension
- The name of the executable code of a framework must have the same name as the framework directory, minus the **.framework** extension. For example, the file name of the executable code for the framework **MyFramework.framework** must be **MyFramework** (or on Windows™, **MyFramework.dll**).
- On Windows™ only: The framework directory must be stored in a directory called **Frameworks**. The executable code (.dll)

must be stored in a directory called **Executables** that has the same parent as the **Frameworks** directory. If the framework is statically linked into applications, the **Executables** directory (full path) should be in the PATH environment variable. (On other OPENSTEP platforms, the executable code file is stored within the framework directory, like a bundle.)

How NSBundles Search for Resources

The algorithm that NSBundles use to find resources has changed somewhat from that of NXBundles, and there are changes to the structure of bundles themselves. A bundle's resources are now stored in a directory call **Resources** within the bundle directory. Within the **Resources** directory are the non-localized resources and the localized resource directories (**English.lproj**, **Swedish.lproj**, etc., as with NXBundle). The addition of the **Resources** directory is primarily a way to simplify Framework versioning. Other changes have been made to reduce the amount of computation required to find resources.

Suppose we are searching for the resource with name **Main** and type **nib**. The bundle's resource path, the "top level", is searched first, for the file **Main.nib**. If the file is not found there, each of the language subdirectories are each searched, in the user's preference order. This is a change from NXBundle's behavior, where the language subdirectories were searched first. This change means that the localized version of a resource that also exists at the top level will not be found, but the top level one will be. All non-localized resources should be placed in the top level, and no localized resources should exist at the top level.

When a resource is found, the algorithm checks to see if a resource of the name **Main-*$(PLATFORM_OS)*** of type **nib** exists. ***$(PLATFORM_OS)*** represents the **make** variable of the same name, and takes on the same values that the **make** variable takes on at compile time. For example, on Windows™, during a project build, ***$(PLATFORM_OS)*** in the **Makefile.postamble** will have the value "winnt". When **Main.nib** is found in a directory, the search algorithm does one final check to see if **Main-winnt.nib** exists in that same directory. If it does, the search algorithm returns the path to that resource; if it does not, the search algorithm returns the path to **Main.nib**. Note that for **Main-winnt.nib** to be found, a file named **Main.nib** must exist in the same directory; this also applies within each language resource directory. It is expected that a developer will choose one of the platform-specific versions to

name without this ***$(PLATFORM_OS)*** suffix, and give the other platform-specific versions of the resource extended names. The values that ***$(PLATFORM_OS)*** can take on are currently "winnt" (on OPENSTEP for Windows™), "nextstep" (on OPENSTEP for MACH), "hpux"(on PDO for HP-UX™), and "solaris" (on PDO for Solaris™).

Another way to accommodate platform-specific resources is by using the **inDirectory:** parameter of the resource-searching methods. The **inDirectory:** parameter is primarily intended to facilitate the collection of resources of a common type or purpose into a single directory; for example, all images could be put into a directory called **Images** within the resource directory. (Note that the AppKit's **+imageNamed:** method in NSImage does not support searching arbitrary directories for images.) As such, the **inDirectory:** parameter can be used to manage platform-specific resources. This is less "automatic" however, and requires the developer to specify the name of the platform-specific directory as well as replicate the required resources and language projects within that directory. The ***$(PLATFORM_OS)*** mechanism described above is simpler to use, particularly if the number of platform-specific resources is small.

Using API not in OpenStep

Many new methods and classes were added for 4.0 and some new methods for 4.1, many outside the purview of the OpenStep specification. To restrict those that are defined to the methods and classes in OpenStep, add the flag **-DSTRICT_OPENSTEP** to the **OTHER_CFLAGS** variable in the project's **Makefile.preamble**.

The Pointer Returned from malloc()

Asking **malloc()** for a block of memory that is close to the size of a page will give you a pointer for which NXZoneFromPtr() returns NULL. This is not a bug; blocks of memory allocated from **malloc()** do not necessarily come from a zone.

Known Problems in OPENSTEP 4.1

This section documents many of the known problems with Foundation in OPENSTEP 4.1.

Reference: 73751

Platforms: All

Problem: Incorrect behavior when copying NSMutableDictionaries.

Description: According to the OpenStep specification, when you send **copy** or **copyWithZone:** to an NSMutableDictionary you should get back a deep immutable copy. This was the behavior implemented in OPENSTEP 4.0. However, in OPENSTEP 4.1 (Windows NT, Mach, and PDO platforms) what you get back is a shallow immutable copyÐthe objects in the dictionary are merely retained instead of being copied.

This change in behavior can have subtle effects within any code that copies NSMutableDictionaries. For example, suppose you have a mutable dictionary populated with objects, and you create a copy (which should be a deep immutable copy). If you then modify one of the objects that you had put into the original dictionary, the modified object will appear in the copy of the dictionary as well as in the original. *This bug means that a copy of a dictionary can change out from under you unexpectedly* (see "Deep vs. Shallow Copies" under General Notes, above, for an explanation of when you can *expect* copies of dictionaries and other collection classes to change out from under you). In an application, you might see no change, incorrect operation, or a crash as a result. Because it doesn't always crash your applications, and because it manifests itself in a number of different ways, this bug can be particularly difficult to detect.

In the next release of OPENSTEP, this behavior will be changed back so that it once again matches the specification. We are currently evaluating a possible patch to correct the behavior in systems running OPENSTEP 4.1.

Workaround: While you may not be able to alter the way that libraries you link against are written, in your own code you can work around this bug simply by avoiding the use of either **copy** or **copyWithZone:** with NSMutableDictionaries. The easiest way to do this is to use the undocumented method **initWithDictionary:copyItems:**, as shown here:

```
[[NSDictionary allocWithZone:NULL] initWithDictionary:original copyItems:YES]
```

initWithDictionary:copyItems: will be public in the next releases of OPENSTEP Enterprise and OPENSTEP for Mach. This method only makes sense when you know you have a dictionary object, but that's often the case (**isKindOfClass:** can be used for testing this). Note that **initWithDictionary:copyItems:** isn't part of the OpenStep specification.

Reference: 47347

Platforms: All

Problem: NSArchiver and NSUnarchiver do not understand bitfields

Description: Structures containing bitfields cannot be archived and unarchived.

Workaround: Encode the members of a structure containing bitfields individually, encoding the bitfields as (unsigned) chars, shorts, or ints.

Reference: 47347

Platforms: All

Problem: NSArchiver and NSUnarchiver do not understand bitfields

Description: Structures containing bitfields cannot be archived and unarchived.

Workaround: Encode the members of a structure containing bitfields individually, encoding the bitfields as (unsigned) chars, shorts, or ints.

Reference: 49084

Platforms: All

Problem: NSInvocation always retains the return value

Description: NSInvocation always retains object return values. This can cause problems if the return value of a method is not a fully initialized object. Only **+alloc** and **+allocWithZone:** return uninitialized objects, so this is only a problem if you are using invocations to send either of those methods to a class object. If you write your own methods which return objects which are not fully initialized, you should avoid sending those messages with an NSInvocation or over Distributed Objects.

Workaround: None

Reference: 56644

Platforms: All

Problem: NSArchiver tracks **char *** pointers as equivalent to their contents

Description: During archiving, NSArchiver keeps track of the **char *** strings it has seen by the address passed in for encoding. If one uses the stack or **malloc()** to construct a **char ***, then encodes it using the "*" type primitive (or @**encode(char *)**), the archiver only remembers the stack/malloc address, assuming it has constant contents.

Workaround: Do not archive **char *** strings from the stack. Do not mutate or free malloc'd **char *** strings during archiving.

Reference: 56659

Platforms: All

Problem: Proxies cannot be notification observers unless they are retained

Description: Because NSNotificationCenter does not retain observers, registering an observer with a remote NSNotificationCenter (the observer would be a proxy in the remote process) will eventually cause a crash of the remote process when it tries to message a freed object, unless the remote process explicitly retains the proxy. This situation is a bit arcane, but the same type of problem can happen with other objects that don't retain objects that they know about.

Workaround: Use an explicit protocol to tell the server (the process vending the notification center) to register and unregister an observer. The server then retains and registers the observer with the vended notification center, and unregisters and releases the observer when you tell it to do so. (If you don't include the unregister half of the protocol, the object will never be released.)

Reference: 57984

Platforms: All

Problem: No way to get list of non-defaults command-line arguments

Description: In NEXTSTEP 3.3 (and earlier releases), command-line arguments that looked like defaults were removed from NXArgv. The array returned by **[[NSProcessInfo processInfo] arguments]**, NXArgv's replacement, however, contains all of the command-line arguments.

Workaround: Code that expected defaults options to have been stripped from NXArgv will have to be rewritten to skip default

option names and their values, or the arguments array will need to be processed to remove them before using it. The **NSArgumentDomain** dictionary can be retrieved from the standard user defaults instance to find out what NSUserDefaults interpreted as a default option.

Reference: 59230

Platforms: Mach, HP-UX™, Solaris™

Problem: The main bundle can be allocated with the wrong directory

Description: If an application is not launched with a full path, NSBundle attempts to create the main bundle instance with the current directory. If the application is launched from the command-line without a full path, and the application is not in the current directory (but found via the shell path), the main bundle will not be created, and the **+mainBundle** method will always return **nil**. This will also be a problem if the application is launched without a full path and changes its current working directory before the **+mainBundle** method is first sent. The Workspace Manager always launches applications as a full path, so this is not a problem for applications launched with Workspace Manager.

Workaround: Launch the application with the full path; launch the app from the directory in which it exists (but not within an **.app** wrapper); create the main bundle instance before changing directories (if applicable); or within the app, change the working directory to the directory in which it lives and allocate the main bundle.

Reference: 61520

Platforms: All

Problem: Handler is called twice when deleting a directory

Description: NSFileManager's **-removeFileAtPath:handler:** method, when deleting a directory, calls the handler method **-fileManager:shouldProceedAfterError:** twice on each subdirectory or file error, and fails to call the handler for the

top-level directory itself.

Workaround: None

Reference: 62235

Platforms: All

Problem: Can't search for characters case-insensitively

Description: NSString's **-rangeOfCharactersFromSet:** ignores the **NSCaseInsensitiveSearch** flag. This also affects NSScanner's **-scanCharactersFromSet:intoString:** method when the scanner has been set to be case insensitive.

Workaround: None

Reference: 62812

Platforms: All

Problem: Changed defaults don't appear sometimes to have taken affect

Description: The default NSUserDefaults domain search order places the NSArgumentDomain before the application's domain. This is usually desirable for reading defaults, but may not be if you set/change default values. If you set a value for a default that was passed as an option on the command-line, the value is correctly set/changed in the application's domain, but **-objectForKey:** will still return the value found in the NSArgumentDomain, because by default it is searched first.

Workaround: There are various workarounds depending upon the application and its use of user defaults. The following options are relative to a particular NSUserDefaults instance (that is, if you create multiple user defaults instances, you'd have to do the workaround for each individually).

1. Remove the NSArgumentDomain from the search list, perhaps after adding any keys it contains to the registration domain; command-line specified defaults will not be found, or not have priority over same-keyed defaults in the application's domain, however.
2. Move the NSArgumentDomain after the application's domain; command-line specified defaults will not have priority over same-keyed defaults in the application's domain, however.
3. Each time you set a default value, set it in the argument domain (difficult and not worth the trouble).
4. Best solution: Create a subclass of NSUserDefaults. For each defaults instance created, create a new volatile domain, tell the instance about it (**-setVolatileDomain:forName:**) perhaps calling it "ChangedDefaults" (don't give it a name beginning with "NS"!), and insert it first in the domain search list of the new defaults instance. Override the **-setObject:forKey:** and **-removeObjectForKey:** methods to set/remove the value in the "ChangedDefaults" volatile domain, and then call **[super ...]** to do the same for the application domain. The following code illustrates what these override methods might look like:

```
- (void)setObject:(id)value forKey:(NSString *)defaultName {
    NSDictionary *changedDomain;
    id oldValue;
    [super setObject:value forKey:defaultName];
    changedDomain = [self volatileDomainForName:@"ChangedDefaults"];
    oldValue = [changedDomain objectForKey:defaultName];
    if (![oldValue isEqual:value]) {     // Something to do
        NSMutableDictionary *newDomain;
        newDomain = [[NSMutableDictionary allocWithZone:[self zone]]
                                        initWithDictionary:changedDomain];
        [newDomain setObject:value forKey:defaultName];
        [self setVolatileDomain:newDomain forName:@"ChangedDefaults"];
        [newDomain release];
```

```
    }
}

- (void)removeObjectForKey:(NSString *)defaultName {
    NSDictionary *changedDomain;
    id oldValue;
    [super removeObjectForKey:defaultName];
    changedDomain = [self volatileDomainForName:@"ChangedDefaults"];
    oldValue = [changedDomain objectForKey:defaultName];
    if (oldValue) {                         // Something to do
        NSMutableDictionary *newDomain;
        newDomain = [[NSMutableDictionary allocWithZone:[self zone]]
                                            initWithDictionary:changedDomain];
        [newDomain removeObjectForKey:defaultName];
        [self setVolatileDomain:newDomain forName:@"ChangedDefaults"];
        [newDomain release];
    }
}
```

Reference: 64605

Platforms: HP-UX™, Solaris™

Problem: NSBundles cannot dynamically load code

Description: Unlike NeXT's implementations of OpenStep on other platforms, Foundation in PDO for HP-UX™ and Solaris™ cannot dynamically load code. This is due to limitations in the compiler and Objective C runtime, which we hope to resolve in a future release.

Workaround: None

Reference: 66411

Platforms: Solaris™

Problem: Formatted floats sometimes loose sign

Description: Formatted strings created with the NSString methods like **+stringWithFormat:** lose the sign of floats formatted with **%f** (and **%g**) in some cases. For instance, `[NSString stringWithFormat:@"%f", -2.3e-2]` results in @"0.023000". The situations in which this occurs have not been fully qualified.

Workaround: Use **sprintf()** to create the character string with the desired format, then create the NSString from that with **+stringWithCString:**.

Reference: 66538

Platforms: All

Problem: NSString **+stringWithFormat:** doesn't work with **%p** or some formatting flags

Description: Unlike what the documentation says, NSString does **not** understand all ANSI C **printf()**-style formatting escapes and flags, and never has.

Workaround: Use **sprintf()** to create the character string with the desired format, then create the NSString from that with **+stringWithCString:**.

Reference: 66766

Platforms: All

Problem: NSBundle doesn't load _profile and _debug binaries in bundles

Description: NSBundle does not attempt to load code from debug or profile versions of their binary code, which the makefiles give the _debug and _profile suffixes, respectively. If the non-debug/non-profile binary doesn't exist in the bundle, no code will be loaded. NSBundle doesn't care how the binary has been compiled, it just only looks for the file with the name given by the **NSExecutable** key in the **Info.plist** file in the **Resources** directory.

Workaround: [Mach, Solaris™, HP-UX™]: Create a symbolic link named without the _profile or _debug suffix to the _profile or _debug binary at the top level of the bundle directory. [All platforms]: Move the _profile or _debug binary to a file of the same name without that suffix. [All platforms]: Modify the value of the **NSExecutable** key in the **Info.plist** file to include the desired _profile or _debug suffix.

Reference: 68449

Platforms: Windows

Problem: Cannot get a proxy for a distributed OLE object using OPENSTEP D.O.

Description: When trying to get a proxy for an OLE object, you may get the following error: "deserializeObjectAt: class `NSDistantIDispatchProxy' not loaded".

Workaround: Include **nxorb.m** (from **NextDeveloper/Libraries**) in your client. Although the *D'OLE Developer's Guide* states that **nxorb.m** is only needed for use with NXConnections, it's needed in some circumstances for NSConnections as well.

Reference: 71118

Platforms: All

Problem: Collections that contain themselves don't describe

Description: Sending the **-description** method to collections which contain themselves, or contain objects that contain the collection, causes infinite recursion.

Workaround: None
