Foundation Reference Update

Cocoa

2007-07-18

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

10.4 Symbol Changes 51

10.3 Symbol Changes 101

10.2 Symbol Changes 127

Introduction to Foundation Reference Update

This document summarizes the symbols that have been added to the Foundation framework. The full reference documentation notes in what version a symbol was introduced, but sometimes it's useful to see only the new symbols for a given release.

If you are not familiar with this framework you should refer to the complete framework reference documentation.

Organization of This Document

Symbols are grouped by class or protocol for Objective-C and by header file for C. For each symbol there is a link to complete documentation, if available, and a brief description, if available.

See Also

For reference documentation on this framework, see *Foundation Framework Reference*.

10.5 Symbol Changes

This article lists the symbols added to `Foundation.framework` in Mac OS X v10.5.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

NSBundle

Complete reference information is available in the `NSBundle` reference.

Instance Methods

`executableArchitectures`	Returns an array of numbers indicating the architecture types supported by the bundle's executable.
`loadAndReturnError:`	Loads the bundle's executable code and returns any errors.
`preflightAndReturnError:`	Returns a Boolean value indicating whether the bundle's executable code could be loaded successfully.
`unload`	Unloads the code associated with the receiver.

NSCalendar

Complete reference information is available in the `NSCalendar` reference.

Class Methods

`autoupdatingCurrentCalendar`	Returns the current logical calendar for the current user.

Instance Methods

`rangeOfUnit:startDate:interval:forDate:`	Returns by reference the starting time and duration of a given calendar unit that contains a given date.

NSCharacterSet

Complete reference information is available in the `NSCharacterSet` reference.

Class Methods

`newlineCharacterSet`	Returns a character set containing the newline characters.

NSCoder

Complete reference information is available in the `NSCoder` reference.

Instance Methods

`decodeIntegerForKey:`	Decodes and returns an NSInteger value that was previously encoded with encodeInt:forKey:, encodeInteger:forKey:, encodeInt32:forKey:, or encodeInt64:forKey: and associated with the string key.
`encodeInteger:forKey:`	Encodes a given NSInteger and associates it with a given key.

NSCondition (New)

Complete reference information is available in the `NSCondition` reference.

Instance Methods

`broadcast`	Signals the condition, waking up all threads waiting on it.
`name`	Returns the name associated with the receiver.
`setName:`	Assigns a name to the receiver.
`signal`	Signals the condition, waking up one thread waiting on it.
`wait`	Blocks the current thread until the condition is signaled.
`waitUntilDate:`	Blocks the current thread until the condition is signaled or the specified time limit is reached.

NSConditionLock

Complete reference information is available in the `NSConditionLock` reference.

Instance Methods

`name`	Returns the name associated with the receiver.
`setName:`	Assigns a name to the receiver.

NSConnection

Complete reference information is available in the `NSConnection` reference.

Class Methods

`serviceConnectionWithName:rootObject:`	Creates and returns a new connection object representing a vended service on the default system port name server.
`serviceConnectionWithName:rootObject: usingNameServer:`	Creates and returns a new connection object representing a vended service on the specified port name server.

NSDateFormatter

Complete reference information is available in the `NSDateFormatter` reference.

Instance Methods

`gregorianStartDate`	Returns the start date of the Gregorian calendar for the receiver.
`longEraSymbols`	Returns the long era symbols for the receiver
`quarterSymbols`	Returns the quarter symbols for the receiver.
`setGregorianStartDate:`	Sets the start date of the Gregorian calendar for the receiver.
`setLongEraSymbols:`	Sets the long era symbols for the receiver.
`setQuarterSymbols:`	Sets the quarter symbols for the receiver.
`setShortQuarterSymbols:`	Sets the short quarter symbols for the receiver.
`setShortStandaloneMonthSymbols:`	Sets the short standalone month symbols for the receiver.
`setShortStandaloneQuarterSymbols:`	Sets the short standalone quarter symbols for the receiver.

`setShortStandaloneWeekdaySymbols:`	Sets the short standalone weekday symbols for the receiver.
`setStandaloneMonthSymbols:`	Sets the standalone month symbols for the receiver.
`setStandaloneQuarterSymbols:`	Sets the standalone quarter symbols for the receiver.
`setStandaloneWeekdaySymbols:`	Sets the standalone weekday symbols for the receiver.
`setVeryShortMonthSymbols:`	Sets the very short month symbols for the receiver.
`setVeryShortStandaloneMonthSymbols:`	Sets the very short standalone month symbols for the receiver.
`setVeryShortStandaloneWeekdaySymbols:`	Sets the very short standalone weekday symbols for the receiver.
`setVeryShortWeekdaySymbols:`	Sets the vert short weekday symbols for the receiver
`shortQuarterSymbols`	Returns the short quarter symbols for the receiver.
`shortStandaloneMonthSymbols`	Returns the short standalone month symbols for the receiver.
`shortStandaloneQuarterSymbols`	Returns the short standalone quarter symbols for the receiver.
`shortStandaloneWeekdaySymbols`	Returns the array of short standalone weekday symbols for the receiver.
`standaloneMonthSymbols`	Returns the standalone month symbols for the receiver.
`standaloneQuarterSymbols`	Returns the standalone quarter symbols for the receiver.
`standaloneWeekdaySymbols`	Returns the array of standalone weekday symbols for the receiver.
`veryShortMonthSymbols`	Returns the very short month symbols for the receiver.
`veryShortStandaloneMonthSymbols`	Returns the very short month symbols for the receiver.
`veryShortStandaloneWeekdaySymbols`	Returns the array of very short standalone weekday symbols for the receiver.
`veryShortWeekdaySymbols`	Returns the array of very short weekday symbols for the receiver.

NSDictionary

Complete reference information is available in the `NSDictionary` reference.

Instance Methods

`getObjects:andKeys:`	Returns by reference C arrays of the keys and values in the receiver.

NSException

Complete reference information is available in the `NSException` reference.

Instance Methods

`callStackReturnAddresses`	Returns the call return addresses related to a raised exception.

NSExpression

Complete reference information is available in the `NSExpression` reference.

Class Methods

`expressionForAggregate:`	Returns a new aggregate expression for a given collection.
`expressionForFunction:selectorName:arguments:`	Returns an expression which will return the result of invoking on a given target a selector with a given name using given arguments.
`expressionForIntersectSet:with:`	Returns a new NSExpression object that represent the intersection of a given set and collection.
`expressionForMinusSet:with:`	Returns a new NSExpression object that represent the subtraction of a given collection from a given set.
`expressionForSubquery:usingIteratorVariable:predicate:`	Returns an expression that filters a collection by storing elements in the collection in a given variable and keeping the elements for which qualifier returns true.
`expressionForUnionSet:with:`	Returns a new NSExpression object that represent the union of a given set and collection.

Instance Methods

`collection`	Returns the collection of expressions in an aggregate expression, or the collection element of a subquery expression.
`leftExpression`	Returns the left expression of an aggregate expression.
`predicate`	Return the predicate of a subquery expression.
`rightExpression`	Returns the right expression of an aggregate expression.

NSFileManager

Complete reference information is available in the `NSFileManager` reference.

Instance Methods

`attributesOfFileSystemForPath:error:`	Returns a dictionary that describes the attributes of the mounted file system on which a given path resides.
`attributesOfItemAtPath:error:`	An NSDictionary object containing the attributes of the item at a given path.
`contentsOfDirectoryAtPath:error:`	Returns an array of NSString objects identifying the directories and files (including symbolic links) contained in a given directory.
`copyItemAtPath:toPath:error:`	Copies the directory or file specified in a given path to a different location in the file system identified by another path.
`createDirectoryAtPath:withIntermediateDirectories: attributes:error:`	Creates a directory with given attributes at a specified path.
`createSymbolicLinkAtPath:withDestinationPath: error:`	Creates a symbolic link identified by a given path that refers to a given location.
`delegate`	Returns the delegate for the receiver.
`destinationOfSymbolicLinkAtPath:error:`	Returns an NSString object containing the path of the item pointed at by the symlink specified by a given path.
`linkItemAtPath:toPath:error:`	Creates a link from a source to a destination.

`moveItemAtPath:toPath:error:`	Moves the directory or file specified by a given path to a different location in the file system identified by another path.
`removeItemAtPath:error:`	Deletes the file, link, or directory (including, recursively, all subdirectories, files, and links in the directory) identified by a given path.
`setAttributes:ofItemAtPath:error:`	Sets the attributes of a given file or directory.
`setDelegate:`	Sets the delegate for the receiver.
`subpathsOfDirectoryAtPath:error:`	Returns an array that contains the filenames of the items in the directory specified by a given path and all its subdirectories recursively.

Delegate Methods

`fileManager:shouldCopyItemAtPath:toPath:`	An NSFileManager object sends this message immediately before attempting to copy to a given path.
`fileManager:shouldLinkItemAtPath:toPath:`	An NSFileManager object sends this message immediately before attempting to link to a given path.
`fileManager:shouldMoveItemAtPath:toPath:`	An NSFileManager object sends this message immediately before attempting to move to a given path.
`fileManager:shouldProceedAfterError:copyingItemAtPath:toPath:`	An NSFileManager object sends this message if an error occurs during an attempt to copy to a given path.
`fileManager:shouldProceedAfterError:linkingItemAtPath:toPath:`	An NSFileManager object sends this message if an error occurs during an attempt to link to a given path.
`fileManager:shouldProceedAfterError:movingItemAtPath:toPath:`	An NSFileManager object sends this message if an error occurs during an attempt to move to a given path.
`fileManager:shouldProceedAfterError:removingItemAtPath:`	An NSFileManager object sends this message if an error occurs during an attempt to delete a given path.

`fileManager:shouldRemoveItemAtPath:`	An NSFileManager object sends this message immediately before attempting to delete an item at a given path.

NSGarbageCollector (New)

Complete reference information is available in the `NSGarbageCollector` reference.

Class Methods

`defaultCollector`	Returns the default garbage collector.

Instance Methods

`collectExhaustively`	Tells the receiver to collect iteratively.
`collectIfNeeded`	Tells the receiver to collect if memory consumption thresholds have been exceeded.
`disable`	Temporarily disables collections.
`disableCollectorForPointer:`	Specifies that a given pointer will not be collected.
`enable`	Enables collection after collection has been disabled.
`enableCollectorForPointer:`	Specifies that a given pointer may be collected.
`isCollecting`	Returns a Boolean value that indicates whether a collection is currently in progress.
`isEnabled`	Returns a Boolean value that indicates whether garbage collection is currently enabled for the current process.
`zone`	Returns a zone of unscanned memory.

NSHashTable (New)

Complete reference information is available in the `NSHashTable` reference.

Class Methods

`hashTableWithOptions:`	Returns a hash table with given pointer functions options.
`hashTableWithWeakObjects`	Returns a new hash table for storing weak references to its contents.

Instance Methods

`addObject:`	Adds a given object to the receiver.
`allObjects`	Returns an array that contains the receiver's members.
`anyObject`	Returns one of the objects in the receiver.
`containsObject:`	Returns a Boolean value that indicates whether the receiver contains a given object.
`count`	Returns the number of elements in the receiver.
`initWithOptions:capacity:`	Returns a hash table initialized with the given attributes.
`initWithPointerFunctions:capacity:`	Returns a hash table initialized with the given functions and capacity.
`intersectHashTable:`	Returns a Boolean value that indicates whether at least one element in the receiver is also present in another given hash table.
`intersectsHashTable:`	Returns a Boolean value that indicates whether a given hash table intersects with the receiver.
`isEqualToHashTable:`	Returns a Boolean value that indicates whether a given hash table is equal to the receiver.
`isSubsetOfHashTable:`	Returns a Boolean value that indicates whether every element in the receiver is also present in another given hash table.
`member:`	Determines whether a given object is an element in the receiver.
`minusHashTable:`	Removes from the receiver each element contained in another given hash table that is present in the receiver.
`objectEnumerator`	Returns an enumerator object that lets you access each object in the receiver.
`pointerFunctions`	Returns the pointer functions for the receiver.
`removeAllObjects`	Removes all objects from the receiver.
`removeObject:`	Removes a given object from the receiver.
`setRepresentation`	Returns a set that contains the receiver's members.
`unionHashTable:`	Adds to the receiver each element contained in another given hash table that is not already a member.

NSIndexSet

Complete reference information is available in the `NSIndexSet` reference.

Instance Methods

`countOfIndexesInRange:`	Returns the number of indexes in the receiver that are members of a given range.

NSInvocationOperation (New)

Complete reference information is available in the `NSInvocationOperation` reference.

Instance Methods

`initWithInvocation:`	Returns an NSInvocationOperation object initialized with the specified invocation object.
`initWithTarget:selector:object:`	Returns an NSInvocationOperation object initialized with the specified target and selector.
`invocation`	Returns the receiver's invocation object.
`result`	Returns the result of the invocation or method.

NSLocale

Complete reference information is available in the `NSLocale` reference.

Class Methods

`autoupdatingCurrentLocale`	Returns the current logical locale for the current user.
`commonISOCurrencyCodes`	Returns an array of common ISO currency codes
`preferredLanguages`	Returns the user's language preference order as an array of strings.

NSLock

Complete reference information is available in the `NSLock` reference.

Instance Methods

`name`	Returns the name associated with the receiver.
`setName:`	Assigns a name to the receiver.

NSMachBootstrapServer

Complete reference information is available in the `NSMachBootstrapServer` reference.

Instance Methods

`servicePortWithName:`	Looks up and returns the port for the vended service that is registered under the specified name.

NSMachPort

Complete reference information is available in the `NSMachPort` reference.

Class Methods

`portWithMachPort:options:`	Creates and returns a port object configured with the specified options and the given Mach port.

Instance Methods

`initWithMachPort:options:`	Initializes a newly allocated NSMachPort object with a given Mach port and the specified options.

NSMapTable (New)

Complete reference information is available in the `NSMapTable` reference.

Class Methods

`mapTableWithKeyOptions:valueOptions:`	Returns a new map table, initialized with the given options
`mapTableWithStrongToStrongObjects`	Returns a new map table object which has strong references to the keys and values.

`mapTableWithStrongToWeakObjects`	Returns a new map table object which has strong references to the keys and weak references to the values.
`mapTableWithWeakToStrongObjects`	Returns a new map table object which has weak references to the keys and strong references to the values.
`mapTableWithWeakToWeakObjects`	Returns a new map table object which has weak references to the keys and values.

Instance Methods

`count`	Returns the number of key-value pairs in the receiver.
`dictionaryRepresentation`	Returns a dictionary representation of the receiver.
`initWithKeyOptions:valueOptions:capacity:`	Returns a map table, initialized with the given options.
`initWithKeyPointerFunctions:valuePointerFunctions:capacity:`	Returns a map table, initialized with the given functions.
`keyEnumerator`	Returns an enumerator object that lets you access each key in the receiver.
`keyPointerFunctions`	Returns the pointer functions the receiver uses to manage keys.
`objectEnumerator`	Returns an enumerator object that lets you access each value in the receiver.
`objectForKey:`	Returns a the value associated with a given key.
`removeAllObjects`	Empties the receiver of its entries.
`removeObjectForKey:`	Removes a given key and its associated value from the receiver.
`setObject:forKey:`	Adds a given key-value pair to the receiver.
`valuePointerFunctions`	Returns the pointer functions the receiver uses to manage values.

NSMethodSignature

Complete reference information is available in the `NSMethodSignature` reference.

Class Methods

`signatureWithObjCTypes:`	Returns an NSMethodSignature object for the given Objective C method type string.

NSMutableSet

Complete reference information is available in the `NSMutableSet` reference.

Instance Methods

`filterUsingPredicate:`	Evaluates a given predicate against the receiver's content and removes from the receiver those objects for which the predicate returns false.

NSNetService

Complete reference information is available in the `NSNetService` reference.

Instance Methods

`port`	Provides the port of the receiver.
`publishWithOptions:`	Attempts to advertise the receiver on the network, with the given options.

NSNumber

Complete reference information is available in the `NSNumber` reference.

Class Methods

`numberWithInteger:`	Creates and returns an NSNumber object containing a given value, treating it as an NSInteger.
`numberWithUnsignedInteger:`	Creates and returns an NSNumber object containing a given value, treating it as an NSUInteger.

Instance Methods

`initWithInteger:`	Returns an NSNumber object initialized to contain a given value, treated as an NSInteger.

`initWithUnsignedInteger:`	Returns an NSNumber object initialized to contain a given value, treated as an NSUInteger.
`integerValue`	Returns the receiver's value as an NSInteger.
`unsignedIntegerValue`	Returns the receiver's value as an NSUInteger.

NSNumberFormatter

Complete reference information is available in the `NSNumberFormatter` reference.

Instance Methods

`currencyGroupingSeparator`	Returns the currency grouping separator for the receiver.
`isLenient`	Returns a Boolean value that indicates whether the receiver uses heuristics to guess at the number which is intended by a string.
`isPartialStringValidationEnabled`	Returns a Boolean value that indicates whether partial string validation is enabled.
`maximumSignificantDigits`	Returns the maximum number of significant digits for the receiver.
`minimumSignificantDigits`	Returns the minimum number of significant digits for the receiver.
`setCurrencyGroupingSeparator:`	Sets the currency grouping separator for the receiver.
`setLenient:`	Sets whether the receiver will use heuristics to guess at the number which is intended by a string.
`setMaximumSignificantDigits:`	Sets the maximum number of significant digits for the receiver.
`setMinimumSignificantDigits:`	Sets the minimum number of significant digits for the receiver.
`setPartialStringValidationEnabled:`	Sets whether partial string validation is enabled for the receiver.
`setUsesSignificantDigits:`	Sets whether the receiver uses significant digits.
`usesSignificantDigits`	Returns a Boolean value that indicates whether the receiver uses significant digits.

NSObject

Complete reference information is available in the `NSObject` reference.

Class Methods

`keyPathsForValuesAffectingValueForKey:`	Returns a set of key paths for properties whose values affect the value of the specified key.
`resolveClassMethod:`	Dynamically provides an implementation for a given selector for a class method.
`resolveInstanceMethod:`	Dynamically provides an implementation for a given selector for an instance method.

Instance Methods

`copyScriptingValue:forKey:withProperties:`	Creates and returns one or more scripting objects to be inserted into the specified relationship by copying the passed-in value and setting the properties in the copied object or objects.
`newScriptingObjectOfClass:forValueForKey:withContentsValue:properties:`	Creates and returns an instance of a scriptable class, setting its contents and properties, for insertion into the relationship identified by the key.
`performSelector:onThread:withObject:waitUntilDone:`	Invokes a method of the receiver on the specified thread using the default mode.
`performSelector:onThread:withObject:waitUntilDone:modes:`	Invokes a method of the receiver on the specified thread using the specified modes.
`performSelectorInBackground:withObject:`	Invokes a method of the receiver on a new background thread.
`scriptingValueForSpecifier:`	Given an object specifier, returns the specified object or objects in the receiving container.

NSOperation (New)

Complete reference information is available in the `NSOperation` reference.

Instance Methods

`addDependency:`	Makes the receiver dependent on the completion of the specified operation.

`cancel`	Advises the operation object that it should stop executing its task.
`dependencies`	Returns a new array object containing the operations on which the receiver is dependent.
`init`	Returns an initialized NSOperation object.
`isCancelled`	Returns a Boolean value indicating whether the operation has been cancelled.
`isConcurrent`	Returns a Boolean value indicating whether the operation runs asynchronously.
`isExecuting`	Returns a Boolean value indicating whether the operation is currently executing.
`isFinished`	Returns a Boolean value indicating whether the operation is done executing.
`isReady`	Returns a Boolean value indicating whether the receiver's operation can be performed now.
`main`	Performs the receiver's non-concurrent task.
`queuePriority`	Returns the priority of the operation in an operation queue.
`removeDependency:`	Removes the receiver's dependence on the specified operation.
`setQueuePriority:`	Sets the priority of the operation when used in an operation queue.
`start`	Begins the execution of the operation.

NSOperationQueue (New)

Complete reference information is available in the `NSOperationQueue` reference.

Instance Methods

`addOperation:`	Adds the specified operation object to the receiver.
`cancelAllOperations`	Cancels all queued and executing operations.
`isSuspended`	Returns a Boolean value indicating whether the receiver is scheduling queued operations for execution.
`maxConcurrentOperationCount`	Returns the maximum number of concurrent operations that the receiver can execute.
`operations`	Returns a new array containing the operations currently in the queue.

`setMaxConcurrentOperationCount:`	Sets the maximum number of concurrent operations that the receiver can execute.
`setSuspended:`	Modifies the execution of pending operations
`waitUntilAllOperationsAreFinished`	Blocks the current thread until all of the receiver's queued and executing operations finish executing.

NSPointerArray (New)

Complete reference information is available in the `NSPointerArray` reference.

Class Methods

`pointerArrayWithOptions:`	Returns a new pointer array initialized to use the given options.
`pointerArrayWithPointerFunctions:`	A new pointer array initialized to use the given functions.
`pointerArrayWithStrongObjects`	Returns a new pointer array that maintains strong references to its elements.
`pointerArrayWithWeakObjects`	Returns a new pointer array that maintains weak references to its elements.

Instance Methods

`addPointer:`	Adds a given pointer to the receiver.
`allObjects`	Returns an array containing all the objects in the receiver.
`compact`	Removes NULL values from the receiver.
`count`	Returns the number of elements in the receiver.
`initWithOptions:`	Initializes the receiver to use the given options.
`initWithPointerFunctions:`	Initializes the receiver to use the given functions.
`insertPointer:atIndex:`	Inserts a pointer at a given index.
`pointerAtIndex:`	Returns the pointer at a given index.
`pointerFunctions`	Returns a new NSPointerFunctions object reflecting the functions in use by the receiver.
`removePointerAtIndex:`	Removes the pointer at a given index.
`replacePointerAtIndex:withPointer:`	Replaces the pointer at a given index.

`setCount:`	Sets the count for the receiver.

NSPointerFunctions (New)

Complete reference information is available in the `NSPointerFunctions` reference.

Class Methods

`pointerFunctionsWithOptions:`	Returns a new NSPointerFunctions object initialized with the given options.

Instance Methods

`acquireFunction`	The function used to acquire memory.
`descriptionFunction`	The function used to describe elements.
`hashFunction`	The hash function.
`initWithOptions:`	Returns an NSPointerFunctions object initialized with the given options.
`isEqualFunction`	The function used to compare pointers.
`relinquishFunction`	The function used to relinquish memory.
`sizeFunction`	The function used to determine the size of pointers.
`usesStrongWriteBarrier`	Specifies whether, in a garbage collected environment, pointers should be assigned using a strong write barrier.
`usesWeakReadAndWriteBarriers`	Specifies whether, in a garbage collected environment, pointers should use weak read and write barriers.

NSPositionalSpecifier

Complete reference information is available in the `NSPositionalSpecifier` reference.

Instance Methods

`objectSpecifier`	Returns the object specifier specified at initialization time.
`position`	Returns the insertion position specified at initialization time.

NSPredicate

Complete reference information is available in the `NSPredicate` reference.

Instance Methods

`evaluateWithObject:substitutionVariables:`	Returns a Boolean value that indicates whether a given object matches the conditions specified by the receiver after substituting in the values in a given variables dictionary.

NSProcessInfo

Complete reference information is available in the `NSProcessInfo` reference.

Instance Methods

`activeProcessorCount`	Provides the number of active processing cores available on the computer.
`physicalMemory`	Provides the amount of physical memory on the computer.
`processorCount`	Provides the number of processing cores available on the computer.

NSProxy

Complete reference information is available in the `NSProxy` reference.

Instance Methods

`finalize`	The garbage collector invokes this method on the receiver before disposing of the memory it uses.

NSRecursiveLock

Complete reference information is available in the `NSRecursiveLock` reference.

Instance Methods

`name`	Returns the name associated with the receiver.
`setName:`	Assigns a name to the receiver

NSRunLoop

Complete reference information is available in the `NSRunLoop` reference.

Class Methods

`mainRunLoop`	Returns the run loop of the main thread.

NSScanner

Complete reference information is available in the `NSScanner` reference.

Instance Methods

`scanHexDouble:`	Scans for a double value from a hexadecimal representation, returning a found value by reference.
`scanHexFloat:`	Scans for a double value from a hexadecimal representation, returning a found value by reference.
`scanHexLongLong:`	Scans for a double value from a hexadecimal representation, returning a found value by reference.
`scanInteger:`	Scans for an NSInteger value from a decimal representation, returning a found value by reference

NSScriptClassDescription

Complete reference information is available in the `NSScriptClassDescription` reference.

Class Methods

`classDescriptionForClass:`	Returns the class description for the specified class or, if it is not scriptable, for the first superclass that is.

Instance Methods

`hasOrderedToManyRelationshipForKey:`	Returns a Boolean value indicating whether the described class has an ordered to-many relationship identified by the specified key.
`hasPropertyForKey:`	Returns a Boolean value indicating whether the described class has a property identified by the specified key.

`hasReadablePropertyForKey:`	Returns a Boolean value indicating whether the described class has a readable property identified by the specified key.
`hasWritablePropertyForKey:`	Returns a Boolean value indicating whether the described class has a writable property identified by the specified key.
`implementationClassName`	Returns the name of the Objective-C class instantiated to implement the scripting class.

NSScriptCommand

Complete reference information is available in the `NSScriptCommand` reference.

Instance Methods

`scriptErrorExpectedTypeDescriptor`	Returns the type descriptor that was put in the reply Apple event if the sender requested a reply, execution of the receiver completed, and an error number was set.
`scriptErrorOffendingObjectDescriptor`	Returns the object descriptor that was put in the reply Apple event if the sender requested a reply, execution of the receiver completed, and an error number was set.
`setScriptErrorExpectedTypeDescriptor:`	Sets a descriptor for the expected type that will be put in the reply Apple event if the sender requested a reply, execution of the receiver completes, and an error number was set.
`setScriptErrorOffendingObjectDescriptor:`	Sets a descriptor for an object that will be put in the reply Apple event if the sender requested a reply, execution of the receiver completes, and an error number was set.

NSScriptObjectSpecifier

Complete reference information is available in the `NSScriptObjectSpecifier` reference.

Class Methods

`objectSpecifierWithDescriptor:`	Returns a new object specifier for an Apple event descriptor.

Instance Methods

`descriptor`	Returns an Apple event descriptor that represents the receiver.

NSSet

Complete reference information is available in the `NSSet` reference.

Instance Methods

`filteredSetUsingPredicate:`	Evaluates a given predicate against each object in the receiver and returns a new set containing the objects for which the predicate returns true.
`setByAddingObject:`	Returns a new set formed by adding a given object to the collection defined by the receiver.
`setByAddingObjectsFromArray:`	Returns a new set formed by adding the objects in a given array to the collection defined by the receiver.
`setByAddingObjectsFromSet:`	Returns a new set formed by adding the objects in a given set to the collection defined by the receiver.

NSSpellServer

Complete reference information is available in the `NSSpellServer` reference.

Delegate Methods

`spellServer:checkGrammarInString:language:details:`	Gives the delegate the opportunity to customize the grammatical analysis of a given string.

NSString

Complete reference information is available in the `NSString` reference.

Instance Methods

`boolValue`	Returns the Boolean value of the receiver's text.

`componentsSeparatedByCharactersInSet:`	Returns an array containing substrings from the receiver that have been divided by characters in a given set.
`getBytes:maxLength:usedLength:encoding: options:range:remainingRange:`	Gets a given range of characters as bytes in a specified encoding.
`integerValue`	Returns the NSInteger value of the receiver's text.
`longLongValue`	Returns the long long value of the receiver's text.
`rangeOfComposedCharacterSequencesForRange:`	Returns the range in the receiver of the composed character sequence in a given range.
`rangeOfString:options:range:locale:`	Finds and returns the range of the first occurrence of a given string within a given range of the receiver, subject to given options, using the specified locale, if any.
`stringByFoldingWithOptions:locale:`	Returns a string with the given character folding options applied.
`stringByReplacingCharactersInRange:withString:`	Returns a new string in which the characters in a specified range of the receiver are replaced by a given string.
`stringByReplacingOccurrencesOfString:withString:`	Returns a new string in which all occurrences of a target string in the receiver are replaced by another given string.
`stringByReplacingOccurrencesOfString:withString: options:range:`	Returns a new string in which all occurrences of a target string in a specified range of the receiver are replaced by another given string.

NSThread

Complete reference information is available in the `NSThread` reference.

Class Methods

`callStackReturnAddresses`	Returns an array containing the call stack return addresses.
`isMainThread`	Returns a Boolean value that indicates whether the current thread is the main thread.

`mainThread`	Returns the NSThread object representing the main thread.
`sleepForTimeInterval:`	Sleeps the thread for a given time interval.

Instance Methods

`cancel`	Changes the cancelled state of the receiver to indicate that it should exit.
`init`	Returns an initialized NSThread object.
`initWithTarget:selector:object:`	Returns an NSThread object initialized with the given arguments.
`isCancelled`	Returns a Boolean value that indicates whether the receiver is cancelled.
`isExecuting`	Returns a Boolean value that indicates whether the receiver is executing.
`isFinished`	Returns a Boolean value that indicates whether the receiver has finished execution.
`isMainThread`	Returns a Boolean value that indicates whether the receiver is the main thread.
`main`	The main entry point routine for the thread.
`name`	Returns the name of the receiver.
`setName:`	Sets the name of the receiver.
`setStackSize:`	Sets the stack size of the receiver.
`stackSize`	Returns the stack size of the receiver.
`start`	Starts the receiver.

NSTimeZone

Complete reference information is available in the `NSTimeZone` reference.

Instance Methods

`daylightSavingTimeOffset`	Returns the current daylight saving time offset of the receiver.
`daylightSavingTimeOffsetForDate:`	Returns the daylight saving time offset for a given date.

`localizedName:locale:`	Returns the name of the receiver localized for a given locale.
`nextDaylightSavingTimeTransition`	Returns the date of the next daylight saving time transition for the receiver.
`nextDaylightSavingTimeTransitionAfterDate:`	Returns the next daylight saving time transition after a given date.

NSURL

Complete reference information is available in the `NSURL` reference.

Class Methods

`fileURLWithPath:isDirectory:`	Initializes and returns a newly created NSURL object as a file URL with a specified path.

Instance Methods

`initFileURLWithPath:isDirectory:`	Initializes a newly created NSURL referencing the local file or directory at path.

NSURLConnection

Complete reference information is available in the `NSURLConnection` reference.

Instance Methods

`initWithRequest:delegate:startImmediately:`	Returns an initialized URL connection and begins to load the data for the URL request, if specified.
`scheduleInRunLoop:forMode:`	Determines the runloop and mode that the receiver uses to send delegate messages to the receiver.
`start`	Causes the receiver to begin loading data, if it has not already.
`unscheduleFromRunLoop:forMode:`	Causes the receiver to stop sending delegate messages using the specified runloop and mode.

NSURLProtocol

Complete reference information is available in the `NSURLProtocol` reference.

Class Methods

`removePropertyForKey:inRequest:`	Removes the property associated with the specified key in the specified request.

NSUserDefaults

Complete reference information is available in the `NSUserDefaults` reference.

Instance Methods

`doubleForKey:`	
`setDouble:forKey:`	

Protocols

All of the protocols with new symbols are listed alphabetically, with their new methods described.

NSFastEnumeration (New)

Complete reference information is available in the `NSFastEnumeration` reference.

Instance Methods

`countByEnumeratingWithState:objects:count:`	Returns by reference a C array of objects over which the sender should iterate, and as the return value the number of objects in the array.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

FoundationErrors.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSExecutableArchitectureMismatchError`	Executable does not provide an architecture compatible with the current process.
`NSExecutableErrorMaximum`	Marks end of the range of error codes reserved for errors related to executable files.
`NSExecutableErrorMinimum`	Marks beginning of the range of error codes reserved for errors related to executable files.
`NSExecutableLinkError`	Executable fails due to linking issues.
`NSExecutableLoadError`	Executable cannot be loaded for some other reason, such as a problem with a library it depends on.
`NSExecutableNotLoadableError`	Executable is of a type that is not loadable in the current process.
`NSExecutableRuntimeMismatchError`	Executable has Objective C runtime information incompatible with the current process.
`NSFileReadTooLargeError`	
`NSFileReadUnknownStringEncodingError`	

NSBundle.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSBundleExecutableArchitectureI386`	Specifies the 32-bit Intel architecture.
`NSBundleExecutableArchitecturePPC`	Specifies the 32-bit PowerPC architecture.
`NSBundleExecutableArchitecturePPC64`	Specifies the 64-bit PowerPC architecture.
`NSBundleExecutableArchitectureX86_64`	Specifies the 64-bit Intel architecture.

NSComparisonPredicate.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSBetweenPredicateOperatorType`	A predicate to determine if the right hand side lies between bounds specified by the left hand side.
`NSContainsPredicateOperatorType`	A predicate to determine if the left hand side contains the right hand side.

NSEnumerator.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFastEnumerationState`	This defines the structure used as contextual information in the NSFastEnumeration protocol.

NSExpression.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSAggregateExpressionType`	An expression that defines an aggregate of NSExpression objects.
`NSIntersectSetExpressionType`	An expression that creates an intersection of the results of two nested expressions.
`NSMinusSetExpressionType`	An expression that combines two nested expression results by set subtraction.
`NSSubqueryExpressionType`	An expression that filters a collection using a subpredicate.
`NSUnionSetExpressionType`	An expression that creates a union of the results of two nested expressions.

NSGeometry.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSPointFromCGPoint`	Returns an NSPoint typecast from a CGPoint.
`NSPointToCGPoint`	Returns a CGPoint typecast from an NSPoint.
`NSRectFromCGRect`	Returns an NSRect typecast from a CGRect.
`NSRectToCGRect`	Returns a CGRect typecast from an NSRect.
`NSSizeFromCGSize`	Returns an NSSize typecast from a CGSize.
`NSSizeToCGSize`	Returns a CGSize typecast from an NSSize.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSGEOMETRY_TYPES_SAME_AS_CGGEOMETRY_TYPES`	
`NSMaxXEdge`	Specifies the right edge of the input rectangle.
`NSMaxYEdge`	Specifies the top edge of the input rectangle.
`NSMinXEdge`	Specifies the left edge of the input rectangle.
`NSMinYEdge`	Specifies the bottom edge of the input rectangle.

NSHashTable.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSHashTableCopyIn`	Equal to NSPointerFunctionsCopyIn.
`NSHashTableObjectPointerPersonality`	Equal to NSPointerFunctionsObjectPointerPersonality.
`NSHashTableOptions`	Type to specify a bit-field used to define the behavior of elements in an NSHashTable object.

`NSHashTableStrongMemory`	Equal to NSPointerFunctionsStrongMemory.
`NSHashTableZeroingWeakMemory`	Equal to NSPointerFunctionsZeroingWeakMemory.
`NSIntegerHashCallBacks`	For sets of NSInteger-sized quantities or smaller (for example, int, long, or unichar).

NSKeyValueObserving.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSKeyValueChangeNotificationIsPriorKey`	
`NSKeyValueObservingOptionInitial`	If specified, a notification should be sent to the observer immediately, before the observer registration method even returns. The change dictionary in the notification will always contain an NSKeyValueChangeNewKey entry if NSKeyValueObservingOptionNew is also specified but will never contain an NSKeyValueChangeOldKey entry. (In an initial notification the current value of the observed property may be old, but it's new to the observer.) You can use this option instead of explicitly invoking, at the same time, code that is also invoked by the observer's observeValueForKeyPath:ofObject: change:context: method. When this option is used withaddObserver:forKeyPath: options:context: a notification will be sent for each indexed object to which the observer is being added.

NSKeyValueObservingOptionPrior	Whether separate notifications should be sent to the observer before and after each change, instead of a single notification after the change. The change dictionary in a notification sent before a change always contains an NSKeyValueChangeNotificationIsPriorKey entry whose value is [NSNumber numberWithBool:YES], but never contains an NSKeyValueChangeNewKey entry. When this option is specified the change dictionary in a notification sent after a change contains the same entries that it would contain if this option were not specified. You can use this option when the observer's own key-value observing-compliance requires it to invoke one of the -willChange... methods for one of its own properties, and the value of that property depends on the value of the observed object's property. (In that situation it's too late to easily invoke -willChange... properly in response to receiving an observeValueForKeyPath:ofObject: change:context: message after the change.)

NSLocale.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSCurrentLocaleDidChangeNotification	Notification that indicates that the user's locale changed.

NSMapTable.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSIntegerMapKeyCallBacks	For keys that are pointer-sized quantities or smaller (for example, int, long, or unichar).
NSIntegerMapValueCallBacks	For values that are pointer-sized quantities, (for example, int, long, or unichar).
NSMapTableCopyIn	Use the memory acquire function to allocate and copy items on input (see acquireFunction [NSPointerFunctions]).
NSMapTableObjectPointerPersonality	Use shifted pointer hash and direct equality, object description.

`NSMapTableOptions`	
`NSMapTableStrongMemory`	Specifies a strong reference from the map table to its contents.
`NSMapTableZeroingWeakMemory`	Specifies a zeroing weak reference from the map table to its contents.
`NSNotAnIntegerMapKey`	Predefined notAKeyMarker for use with NSMapTableKeyCallBacks.

NSNetServices.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSNetServiceNoAutoRename`	Specifies that the network service not rename itself in the event of a name collision.
`NSNetServiceOptions`	These constants specify options for a network service.

NSObjCRuntime.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSProtocolFromString`	Returns a the protocol with a given name.
`NSStringFromProtocol`	Returns the name of a protocol as a string.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NS_INLINE`	
`NS_REQUIRES_NIL_TERMINATION`	
`NSFoundationVersionNumber10_1_1`	Foundation version released in Mac OS X version 10.1.1.

NSFoundationVersionNumber10_1_2	Foundation version released in Mac OS X version 10.1.2.
NSFoundationVersionNumber10_1_3	Foundation version released in Mac OS X version 10.1.3.
NSFoundationVersionNumber10_1_4	Foundation version released in Mac OS X version 10.1.4.
NSFoundationVersionNumber10_2_1	Foundation version released in Mac OS X version 10.2.1.
NSFoundationVersionNumber10_2_2	Foundation version released in Mac OS X version 10.2.2.
NSFoundationVersionNumber10_2_3	Foundation version released in Mac OS X version 10.2.3.
NSFoundationVersionNumber10_2_4	Foundation version released in Mac OS X version 10.2.4.
NSFoundationVersionNumber10_2_5	Foundation version released in Mac OS X version 10.2.5.
NSFoundationVersionNumber10_2_6	Foundation version released in Mac OS X version 10.2.6.
NSFoundationVersionNumber10_2_7	Foundation version released in Mac OS X version 10.2.7.
NSFoundationVersionNumber10_2_8	Foundation version released in Mac OS X version 10.2.8.
NSFoundationVersionNumber10_3_1	Foundation version released in Mac OS X version 10.3.1.
NSFoundationVersionNumber10_3_5	Foundation version released in Mac OS X version 10.3.5.
NSFoundationVersionNumber10_3_6	Foundation version released in Mac OS X version 10.3.6.
NSFoundationVersionNumber10_3_7	Foundation version released in Mac OS X version 10.3.7.
NSFoundationVersionNumber10_3_8	Foundation version released in Mac OS X version 10.3.8.
NSFoundationVersionNumber10_3_9	Foundation version released in Mac OS X version 10.3.9.
NSFoundationVersionNumber10_4	Foundation version released in Mac OS X version 10.4.
NSFoundationVersionNumber10_4_1	Foundation version released in Mac OS X version 10.4.1.

NSFoundationVersionNumber10_4_10	Foundation version released in Mac OS X version 10.4.10.
NSFoundationVersionNumber10_4_11	Foundation version released in Mac OS X version 10.4.11.
NSFoundationVersionNumber10_4_2	Foundation version released in Mac OS X version 10.4.2.
NSFoundationVersionNumber10_4_3	Foundation version released in Mac OS X version 10.4.3.
NSFoundationVersionNumber10_4_4_Intel	Foundation version released in Mac OS X version 10.4.4 for Intel.
NSFoundationVersionNumber10_4_4_PowerPC	Foundation version released in Mac OS X version 10.4.4 for PowerPC.
NSFoundationVersionNumber10_4_5	Foundation version released in Mac OS X version 10.4.5.
NSFoundationVersionNumber10_4_6	Foundation version released in Mac OS X version 10.4.6.
NSFoundationVersionNumber10_4_7	Foundation version released in Mac OS X version 10.4.7.
NSFoundationVersionNumber10_4_8	Foundation version released in Mac OS X version 10.4.8.
NSFoundationVersionNumber10_4_9	Foundation version released in Mac OS X version 10.4.9.
NSInteger	Used to describe an integer.
NSINTEGER_DEFINED	
NSIntegerMax	The maximum value for an NSInteger.
NSIntegerMin	The minimum value for an NSInteger.
NSUInteger	Used to describe an unsigned integer.
NSUIntegerMax	The maximum value for an NSUInteger.

NSOperation.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSInvocationOperationCancelledException`	The name of the exception raised if the result method is called after the operation was cancelled.
`NSInvocationOperationVoidResultException`	The name of the exception raised if the result method is called for an invocation method with a void return type.
`NSOperationQueueDefaultMaxConcurrentOperationCount`	The default maximum number of operations is determined dynamically by the NSOperationQueue object based on current system conditions.
`NSOperationQueuePriority`	Describes the priority of an operation relative to other operations in an operation queue.
`NSOperationQueuePriorityHigh`	Operations receive high priority for execution.
`NSOperationQueuePriorityLow`	Operations receive low priority for execution.
`NSOperationQueuePriorityNormal`	Operations receive the normal priority for execution.
`NSOperationQueuePriorityVeryHigh`	Operations receive very high priority for execution.
`NSOperationQueuePriorityVeryLow`	Operations receive very low priority for execution.

NSPathUtilities.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSDownloadsDirectory`	Location of the user's downloads directory.

NSPointerFunctions.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSPointerFunctionsCopyIn`	Use the memory acquire function to allocate and copy items on input (see acquireFunction).
`NSPointerFunctionsCStringPersonality`	Use a string hash and strcmp; C-string '%s' style description.
`NSPointerFunctionsIntegerPersonality`	Use unshifted value as hash and equality.
`NSPointerFunctionsMachVirtualMemory`	Use Mach memory.
`NSPointerFunctionsMallocMemory`	Use free() on removal, calloc() on copy in.
`NSPointerFunctionsObjectPersonality`	Use hash and isEqual methods for hashing and equality comparisons, use the description method for a description.
`NSPointerFunctionsObjectPointerPersonality`	Use shifted pointer for the hash value and direct comparison to determine equality; use the description method for a description.
`NSPointerFunctionsOpaqueMemory`	Take no action when pointers are deleted.
`NSPointerFunctionsOpaquePersonality`	Use shifted pointer for the hash value and direct comparison to determine equality.
`NSPointerFunctionsOptions`	Defines the memory and personality options for an NSPointerFunctions object.
`NSPointerFunctionsStrongMemory`	Use strong write-barriers to backing store; use garbage-collected memory on copy-in.
`NSPointerFunctionsStructPersonality`	Use a memory hash and memcmp (using a size function that you must set—see sizeFunction).
`NSPointerFunctionsZeroingWeakMemory`	Use weak read and write barriers; use garbage-collected memory on copyIn.

NSPort.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSMachPortDeallocateNone`	Do not remove any send or receive rights.
`NSMachPortDeallocateReceiveRight`	Remove a receive right when the NSMachPort object is invalidated or destroyed.
`NSMachPortDeallocateSendRight`	Deallocate a send right when the NSMachPort object is invalidated or destroyed.

NSRunLoop.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSRunLoopCommonModes`	Objects added to a run loop using this value as the mode are monitored by all run loop modes that have been declared as a member of the set of “common" modes; see the description of CFRunLoopAddCommonMode for details.

NSSpellServer.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSGrammarCorrections`	
`NSGrammarRange`	
`NSGrammarUserDescription`	

NSString.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSDiacriticInsensitiveSearch`	Search ignores diacritic marks.
`NSForcedOrderingSearch`	Comparisons are forced to return either NSOrderedAscending or NSOrderedDescending if the strings are equivalent but not strictly equal.
`NSStringCompareOptions`	Type for string comparison options.
`NSStringEncodingConversionAllowLossy`	Allows lossy conversion.
`NSStringEncodingConversionExternalRepresentation`	
`NSStringEncodingConversionOptions`	Type for encoding conversion options.

NSUTF16BigEndianStringEncoding	NSUTF16StringEncoding encoding with explicit endianness specified.
NSUTF16LittleEndianStringEncoding	NSUTF16StringEncoding encoding with explicit endianness specified.
NSUTF16StringEncoding	
NSUTF32BigEndianStringEncoding	NSUTF32StringEncoding encoding with explicit endianness specified.
NSUTF32LittleEndianStringEncoding	NSUTF32StringEncoding encoding with explicit endianness specified.
NSUTF32StringEncoding	32-bit UTF encoding.
NSWidthInsensitiveSearch	Search ignores width differences in characters that have full-width and half-width forms, as occurs in East Asian character sets.

NSTimeZone.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSSystemTimeZoneDidChangeNotification	Sent when the time zone changed.
NSTimeZoneNameStyle	Defines a type for time zone name styles.
NSTimeZoneNameStyleDaylightSaving	Specifies a daylight saving name style.
NSTimeZoneNameStyleShortDaylightSaving	Specifies a short daylight saving name style.
NSTimeZoneNameStyleShortStandard	Specifies a short name style.
NSTimeZoneNameStyleStandard	Specifies a standard name style.

NSURLError.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSURLErrorCannotDecodeContentData	

NSURLErrorCannotDecodeRawData	
NSURLErrorCannotParseResponse	
NSURLErrorDataLengthExceedsMaximum	Returned when the length of the resource data is too exceeds the maximum allowed.

NSURLRequest.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSURLRequestReloadIgnoringLocalAndRemoteCacheData	Specifies that not only should the local cache data be ignored, but that proxies and other intermediates should be instructed to disregard their caches so far as the protocol allows.
NSURLRequestReloadIgnoringLocalCacheData	Specifies that the data for the URL load should be loaded from the originating source. No existing cache data should be used to satisfy a URL load request.
NSURLRequestReloadRevalidatingCacheData	Specifies that the existing cache data may be used provided the origin source confirms its validity, otherwise the URL is loaded from the origin source.

NSValueTransformer.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSKeyedUnarchiveFromDataTransformerName	This value transformer returns an object created by attempting to unarchive the data in the NSData object passed as the value. The archived object must be created using keyed archiving in order to be unarchived and archived with this transformer.

NSZone.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSMakeCollectable`	Makes a newly allocated Core Foundation object eligible for collection.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSCollectorDisabledOption`	Provides an allocation for use as an external reference.

10.4 Symbol Changes

This article lists the symbols added to `Foundation.framework` in Mac OS X v10.4.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

NSAffineTransform (New)

Complete reference information is available in the `NSAffineTransform` reference.

Class Methods

`transform`	Creates and returns a new NSAffineTransform object initialized to the identity matrix.

Instance Methods

`appendTransform:`	Appends the specified matrix to the receiver's matrix.
`initWithTransform:`	Initializes the receiver's matrix using another transform object and returns the receiver.
`invert`	Replaces the receiver's matrix with its inverse matrix.
`prependTransform:`	Prepends the specified matrix to the receiver's matrix.
`rotateByDegrees:`	Applies a rotation factor (measured in degrees) to the receiver's transformation matrix.
`rotateByRadians:`	Applies a rotation factor (measured in radians) to the receiver's transformation matrix.
`scaleBy:`	Applies the specified scaling factor along both x and y axes to the receiver's transformation matrix.
`scaleXBy:yBy:`	Applies scaling factors to each axis of the receiver's transformation matrix.

`setTransformStruct:`	Replaces the receiver's transformation matrix with the specified values.
`transformPoint:`	Applies the receiver's transform to the specified NSPoint data type and returns the results.
`transformSize:`	Applies the receiver's transform to the specified NSSize data type and returns the results.
`transformStruct`	Returns the matrix coefficients stored in the receiver's matrix.
`translateXBy:yBy:`	Applies the specified translation factors to the receiver's transformation matrix.

NSArray

Complete reference information is available in the `NSArray` reference.

Instance Methods

`addObserver:forKeyPath:options:context:`	Raises an exception.
`filteredArrayUsingPredicate:`	Evaluates a given predicate against each object in the receiver and returns a new array containing the objects for which the predicate returns true.
`objectsAtIndexes:`	Returns an array containing the objects in the receiver at the indexes specified by a given index set.
`removeObserver:forKeyPath:`	Raises an exception.

NSAutoreleasePool

Complete reference information is available in the `NSAutoreleasePool` reference.

Instance Methods

`drain`	In a reference-counted environment, releases and pops the receiver; in a garbage-collected environment, triggers garbage collection if the memory allocated since the last collection is greater than the current threshold.

NSCalendar (New)

Complete reference information is available in the `NSCalendar` reference.

Class Methods

`currentCalendar`	Returns the logical calendar for the current user.

Instance Methods

`calendarIdentifier`	Returns the identifier for the receiver.
`components:fromDate:`	Returns a NSDateComponents object containing a given date decomposed into specified components.
`components:fromDate:toDate:options:`	Returns, as an NSDateComponents object using specified components, the difference between two supplied dates.
`dateByAddingComponents:toDate:options:`	Returns a new NSDate object representing the absolute time calculated by adding given components to a given date.
`dateFromComponents:`	Returns a new NSDate object representing the absolute time calculated from given components.
`firstWeekday`	Returns the index of the first weekday of the receiver.
`initWithCalendarIdentifier:`	Initializes a newly-allocated NSCalendar object for the calendar specified by a given identifier.
`locale`	Returns the locale for the receiver.
`maximumRangeOfUnit:`	The maximum range limits of the values that a given unit can take on in the receive
`minimumDaysInFirstWeek`	Returns the minimum number of days in the first week of the receiver.
`minimumRangeOfUnit:`	Returns the minimum range limits of the values that a given unit can take on in the receiver.
`ordinalityOfUnit:inUnit:forDate:`	Returns, for a given absolute time, the ordinal number of a smaller calendar unit (such as a day) within a specified larger calendar unit (such as a week).
`rangeOfUnit:inUnit:forDate:`	Returns the range of absolute time values that a smaller calendar unit (such as a day) can take on in a larger calendar unit (such as a month) that includes a specified absolute time.
`setFirstWeekday:`	Sets the index of the first weekday for the receiver.
`setLocale:`	Sets the locale for the receiver.
`setMinimumDaysInFirstWeek:`	Sets the minimum number of days in the first week of the receiver.

`setTimeZone:`	Sets the time zone for the receiver.
`timeZone`	Returns the time zone for the receiver.

NSComparisonPredicate (New)

Complete reference information is available in the `NSComparisonPredicate` reference.

Class Methods

`predicateWithLeftExpression:rightExpression:customSelector:`	Returns a new predicate formed by combining the left and right expressions using a given selector.
`predicateWithLeftExpression:rightExpression:modifier:type:options:`	Creates and returns a predicate of a given type formed by combining given left and right expressions using a given modifier and options.

Instance Methods

`comparisonPredicateModifier`	Returns the comparison predicate modifier for the receiver.
`customSelector`	Returns the selector for the receiver.
`initWithLeftExpression:rightExpression:customSelector:`	Initializes a predicate formed by combining given left and right expressions using a given selector.
`initWithLeftExpression:rightExpression:modifier:type:options:`	Initializes a predicate to a given type formed by combining given left and right expressions using a given modifier and options.
`leftExpression`	Returns the left expression for the receiver.
`options`	Returns the options that are set for the receiver.
`predicateOperatorType`	Returns the predicate type for the receiver.
`rightExpression`	Returns the right expression for the receiver.

NSCompoundPredicate (New)

Complete reference information is available in the `NSCompoundPredicate` reference.

Class Methods

`andPredicateWithSubpredicates:`	Returns a new predicate formed by AND-ing the predicates in a given array.
`notPredicateWithSubpredicate:`	Returns a new predicate formed by NOT-ing a given predicate.
`orPredicateWithSubpredicates:`	Returns a new predicate formed by OR-ing the predicates in a given array.

Instance Methods

`compoundPredicateType`	Returns the predicate type for the receiver.
`initWithType:subpredicates:`	Returns the receiver initialized to a given type using predicates from a given array.
`subpredicates`	Returns the array of the receiver's subpredicates.

NSData

Complete reference information is available in the `NSData` reference.

Class Methods

`dataWithContentsOfFile:options:error:`	Creates and returns a data object by reading every byte from the file specified by a given path.
`dataWithContentsOfURL:options:error:`	Creates and returns a data object containing the data from the location specified by aURL.

Instance Methods

`initWithContentsOfFile:options:error:`	Returns a data object initialized by reading into it the data from the file specified by a given path.
`initWithContentsOfURL:options:error:`	Returns a data object initialized with the data from the location specified by a given URL.
`writeToFile:options:error:`	Writes the bytes in the receiver to the file specified by a given path.
`writeToURL:options:error:`	Writes the bytes in the receiver to the location specified by a given URL.

NSDateComponents (New)

Complete reference information is available in the `NSDateComponents` reference.

Instance Methods

`day`	Returns the number of day units for the receiver.
`era`	Returns the number of era units for the receiver.
`hour`	Returns the number of hour units for the receiver.
`minute`	Returns the number of minute units for the receiver.
`month`	Returns the number of month units for the receiver.
`second`	Returns the number of second units for the receiver.
`setDay:`	Sets the number of day units for the receiver.
`setEra:`	Sets the number of era units for the receiver.
`setHour:`	Sets the number of hour units for the receiver.
`setMinute:`	Sets the number of minute units for the receiver.
`setMonth:`	Sets the number of month units for the receiver.
`setSecond:`	Sets the number of second units for the receiver.
`setWeek:`	Sets the number of week units for the receiver.
`setWeekday:`	Sets the number of weekday units for the receiver.
`setWeekdayOrdinal:`	Sets the ordinal number of weekday units for the receiver.
`setYear:`	Sets the number of year units for the receiver.
`week`	Returns the number of week units for the receiver.
`weekday`	Returns the number of weekday units for the receiver.
`weekdayOrdinal`	Returns the ordinal number of weekday units for the receiver.
`year`	Returns the number of year units for the receiver.

NSDateFormatter

Complete reference information is available in the `NSDateFormatter` reference.

Class Methods

`defaultFormatterBehavior`	Returns the default formatting behavior for instances of the class.
`setDefaultFormatterBehavior:`	Sets the default formatting behavior for instances of the class.

Instance Methods

`AMSymbol`	Returns the AM symbol for the receiver.
`calendar`	Returns the calendar for the receiver.
`dateFromString:`	Returns a date representation of a given string interpreted using the receiver's current settings.
`dateStyle`	Returns the date style of the receiver.
`defaultDate`	Returns the default date for the receiver.
`eraSymbols`	Returns the era symbols for the receiver.
`formatterBehavior`	Returns the formatter behavior for the receiver.
`generatesCalendarDates`	Returns a Boolean value that indicates whether the receiver generates calendar dates.
`getObjectValue:forString:range:error:`	Returns by reference a date representation of a given string and the range of the string used, and returns a Boolean value that indicates whether the string could be parsed.
`init`	Initializes and returns an NSDateFormatter instance.
`isLenient`	Returns a Boolean value that indicates whether the receiver uses heuristics when parsing a string.
`locale`	Returns the locale for the receiver.
`monthSymbols`	Returns the month symbols for the receiver.
`PMSymbol`	Returns the PM symbol for the receiver.
`setAMSymbol:`	Sets the AM symbol for the receiver.
`setCalendar:`	Sets the calendar for the receiver.
`setDateFormat:`	Sets the date format for the receiver.
`setDateStyle:`	Sets the date style of the receiver.
`setDefaultDate:`	Sets the default date for the receiver.
`setEraSymbols:`	Sets the era symbols for the receiver.

`setFormatterBehavior:`	Sets the formatter behavior for the receiver.
`setGeneratesCalendarDates:`	Sets whether the receiver generates calendar dates.
`setLenient:`	Sets whether the receiver uses heuristics when parsing a string.
`setLocale:`	Sets the locale for the receiver.
`setMonthSymbols:`	Sets the month symbols for the receiver.
`setPMSymbol:`	Sets the PM symbol for the receiver.
`setShortMonthSymbols:`	Sets the short month symbols for the receiver.
`setShortWeekdaySymbols:`	Sets the short weekday symbols for the receiver.
`setTimeStyle:`	Sets the time style of the receiver.
`setTimeZone:`	Sets the time zone for the receiver.
`setTwoDigitStartDate:`	Sets the two-digit start date for the receiver.
`setWeekdaySymbols:`	Sets the weekday symbols for the receiver.
`shortMonthSymbols`	Returns the array of short month symbols for the receiver.
`shortWeekdaySymbols`	Returns the array of short weekday symbols for the receiver.
`stringFromDate:`	Returns a string representation of a given date formatted using the receiver's current settings.
`timeStyle`	Returns the time style of the receiver.
`timeZone`	Returns the time zone for the receiver.
`twoDigitStartDate`	Returns the earliest date that can be denoted by a two-digit year specifier.
`weekdaySymbols`	Returns the array of weekday symbols for the receiver.

NSError

Complete reference information is available in the `NSError` reference.

Instance Methods

`localizedFailureReason`	Returns a string containing the localized explanation of the reason for the error.

`localizedRecoveryOptions`	Returns an array containing the localized titles of buttons appropriate for displaying in an alert panel.
`localizedRecoverySuggestion`	Returns a string containing the localized recovery suggestion for the error.
`recoveryAttempter`	Returns an object that conforms to the NSErrorRecoveryAttempting informal protocol.

Delegate Methods

`attemptRecoveryFromError:optionIndex:`	Implemented to attempt a recovery from an error noted in an application-modal dialog.
`attemptRecoveryFromError:optionIndex:delegate:didRecoverSelector:contextInfo:`	Implemented to attempt a recovery from an error noted in an document-modal sheet.

NSExpression (New)

Complete reference information is available in the `NSExpression` reference.

Class Methods

`expressionForConstantValue:`	Returns a new expression that represents a given constant value.
`expressionForEvaluatedObject`	Returns a new expression that represents the object being evaluated.
`expressionForFunction:arguments:`	Returns a new expression that will invoke one of the predefined functions.
`expressionForKeyPath:`	Returns a new expression that invokes valueForKeyPath: with a given key path.
`expressionForVariable:`	Returns a new expression that extracts a value from the variable bindings dictionary for a given key.

Instance Methods

`arguments`	Returns the arguments for the receiver.
`constantValue`	Returns the constant value of the receiver.
`expressionType`	Returns the expression type for the receiver.
`expressionValueWithObject:context:`	Evaluates an expression using a given object and context.

`function`	Returns the function for the receiver.
`initWithExpressionType:`	Initializes the receiver with the specified expression type.
`keyPath`	Returns the key path for the receiver.
`operand`	Returns the operand for the receiver.
`variable`	Returns the variable for the receiver.

NSIndexPath (New)

Complete reference information is available in the `NSIndexPath` reference.

Class Methods

`indexPathWithIndex:`	Creates an one-node index path.
`indexPathWithIndexes:length:`	Creates an index path with one or more nodes.

Instance Methods

`compare:`	Indicates the depth-first traversal order of the receiver and another index path.
`getIndexes:`	Provides a reference to the receiver's indexes.
`indexAtPosition:`	Provides the index at a particular node in the receiver.
`indexPathByAddingIndex:`	Provides an index path containing the indexes in the receiver and another index.
`indexPathByRemovingLastIndex`	Provides an index path with the indexes in the receiver, excluding the last one.
`initWithIndex:`	Initializes an allocated NSIndexPath object with a one-node index path.
`initWithIndexes:length:`	Initializes an allocated NSIndexPath object with an index path of a specific length.
`length`	Provides the number of indexes in the receiver.

NSLocale (New)

Complete reference information is available in the `NSLocale` reference.

Class Methods

`availableLocaleIdentifiers`	Returns an array of NSString objects, each of which identifies a locale available on the system.
`canonicalLocaleIdentifierFromString:`	Returns the canonical identifier for a given locale identification string.
`componentsFromLocaleIdentifier:`	Returns a dictionary that is the result of parsing a locale ID.
`currentLocale`	Returns the logical locale for the current user.
`ISOCountryCodes`	Returns an array of NSString objects that represents all known legal country codes.
`ISOCurrencyCodes`	Returns an array of NSString objects that represents all known legal ISO currency codes.
`ISOLanguageCodes`	Returns an array of NSString objects that represents all known legal ISO language codes.
`localeIdentifierFromComponents:`	Returns a locale identifier from the components specified in a given dictionary.
`systemLocale`	Returns the "root", canonical locale, that contains fixed "backstop" settings that provide values for otherwise undefined keys.

Instance Methods

`displayNameForKey:value:`	Returns the display name for the given value.
`initWithLocaleIdentifier:`	Initializes the receiver using a given locale identifier.
`localeIdentifier`	Returns the identifier for the receiver.
`objectForKey:`	Returns the object corresponding to the specified key.

NSMetadataItem (New)

Complete reference information is available in the `NSMetadataItem` reference.

Instance Methods

`attributes`	Returns an array containing the attribute names of the receiver's values.
`valueForAttribute:`	Returns the receiver's metadata attribute name specified by a given key.

`valuesForAttributes:`	Returns a dictionary containing the key-value pairs for the attribute names specified by a given array of keys.

NSMetadataQuery (New)

Complete reference information is available in the `NSMetadataQuery` reference.

Instance Methods

`delegate`	Returns the receiver's delegate.
`disableUpdates`	Disables updates to the query results.
`enableUpdates`	Enables updates to the query results.
`groupedResults`	Returns an array containing hierarchical groups of query results based on the receiver's grouping attributes.
`groupingAttributes`	Returns the receiver's grouping attributes.
`indexOfResult:`	Returns the index of a query result object in the receiver's results array.
`init`	Initializes an allocated NSMetadataQuery object.
`isGathering`	Returns a Boolean value that indicates whether the receiver is in the initial gathering phase of the query.
`isStarted`	Returns a Boolean value that indicates whether the receiver has started the query.
`isStopped`	Returns a Boolean value that indicates whether the receiver has stopped the query.
`notificationBatchingInterval`	Returns the interval that the receiver provides notification of updated query results.
`predicate`	Returns the predicate the receiver uses to filter query results.
`resultAtIndex:`	Returns the query result at a specific index.
`resultCount`	Returns the number of results returned by the receiver.
`results`	Returns an array containing the result objects for the receiver.
`searchScopes`	Returns an array containing the receiver's search scopes.
`setDelegate:`	Sets the receiver's delegate
`setGroupingAttributes:`	Sets the receiver's grouping attributes to specific attribute names.

`setNotificationBatchingInterval:`	Sets the interval between update notifications sent by the receiver.
`setPredicate:`	Sets the predicate used by the receiver to filter the query results.
`setSearchScopes:`	Resctrict the search scope of the receiver.
`setSortDescriptors:`	Sets the sort descriptors to be used by the receiver.
`setValueListAttributes:`	Sets the value list attributes for the receiver to the specific attribute names.
`sortDescriptors`	Returns an array containing the receiver's sort descriptors.
`startQuery`	Attempts to start the query.
`stopQuery`	Stops the receiver's current query from gathering any further results.
`valueListAttributes`	Returns an array containing the value list attributes the receiver generates.
`valueLists`	Returns a dictionary containing the value lists generated by the receiver.
`valueOfAttribute:forResultAtIndex:`	Returns the value for the attribute name attrName at the index in the results specified by idx.

Delegate Methods

`metadataQuery:replacementObjectForResultObject:`	Implemented by the delegate to return a different object for a specific query result object.
`metadataQuery:replacementValueForAttribute:value:`	Implemented by the delegate to return a different value for a specific attribute.

NSMetadataQueryAttributeValueTuple (New)

Complete reference information is available in the `NSMetadataQueryAttributeValueTuple` reference.

Instance Methods

`attribute`	Returns the receiver's attribute name.
`count`	Returns the number of instances of the value that exist for the attribute name of the receiver.
`value`	Returns the receiver's attribute value.

NSMetadataQueryResultGroup (New)

Complete reference information is available in the `NSMetadataQueryResultGroup` reference.

Instance Methods

`attribute`	Returns the attribute name for the receiver's result group.
`resultAtIndex:`	Returns the query result at a specific index.
`resultCount`	Returns the number of results returned by the receiver.
`results`	Returns an array containing the result objects for the receiver.
`subgroups`	Returns an array containing the subgroups of the receiver.
`value`	Returns the value of the attribute name for the receiver.

NSMutableArray

Complete reference information is available in the `NSMutableArray` reference.

Instance Methods

`filterUsingPredicate:`	Evaluates a given predicate against the receiver's content and leaves only objects that match
`insertObjects:atIndexes:`	Inserts the objects in in a given array into the receiver at the specified indexes.
`removeObjectsAtIndexes:`	Removes the objects at the specified indexes from the receiver.
`replaceObjectsAtIndexes:withObjects:`	Replaces the objects in the receiver at specified locations specified with the objects from a given array.

NSMutableURLRequest

Complete reference information is available in the `NSMutableURLRequest` reference.

Instance Methods

`setHTTPBodyStream:`	Sets the request body of the receiver to the contents of a specified input stream.

NSNetService

Complete reference information is available in the `NSNetService` reference.

Class Methods

`dataFromTXTRecordDictionary:`	Returns an NSData object representing a TXT record formed from a given dictionary.
`dictionaryFromTXTRecordData:`	Returns a dictionary representing a TXT record given as an NSData object.

Instance Methods

`getInputStream:outputStream:`	Retrieves by reference the input and output streams for the receiver and returns a Boolean value that indicates whether they were retrieved successfully.
`hostName`	Returns the host name of the computer providing the service.
`resolveWithTimeout:`	Starts a resolve process of a finite duration for the receiver.
`setTXTRecordData:`	Sets the TXT record for the receiver, and returns a Boolean value that indicates whether the operation was successful.
`startMonitoring`	Starts the monitoring of TXT-record updates for the receiver.
`stopMonitoring`	Stops the monitoring of TXT-record updates for the receiver.
`TXTRecordData`	Returns the TXT record for the receiver.

Delegate Methods

`netService:didUpdateTXTRecordData:`	Notifies the delegate that the TXT record for a given service has been updated.
`netServiceDidPublish:`	Notifies the delegate that a service was successfully published.

NSNetServiceBrowser

Complete reference information is available in the `NSNetServiceBrowser` reference.

Instance Methods

`searchForBrowsableDomains`	Initiates a search for domains visible to the host. This method returns immediately.

NSNumberFormatter

Complete reference information is available in the `NSNumberFormatter` reference.

Class Methods

`defaultFormatterBehavior`	Returns an NSNumberFormatterBehavior constant that indicates default formatter behavior for new instances of NSNumberFormatter.
`setDefaultFormatterBehavior:`	Sets the default formatter behavior for new instances of NSNumberFormatter .

Instance Methods

`alwaysShowsDecimalSeparator`	Returns a Boolean value that indicates whether the receiver always shows a decimal separator, even if the number is an integer.
`currencyCode`	Returns the receiver's currency code as a string.
`currencyDecimalSeparator`	Returns the receiver's currency decimal separator as a string.
`currencySymbol`	Returns the receiver's local currency symbol.
`exponentSymbol`	Returns the string the receiver uses as an exponent symbol.
`formatterBehavior`	Returns an NSNumberFormatterBehavior constant that indicates the formatter behavior of the receiver.
`formatWidth`	Returns the format width of the receiver.
`generatesDecimalNumbers`	Returns a Boolean value that indicates whether the receiver creates instances of NSDecimalNumber when it converts strings to number objects.
`getObjectValue:forString:range:error:`	Returns by reference a cell-content object after creating it from a range of characters in a given string.
`groupingSeparator`	Returns a string containing the receiver's grouping separator.
`groupingSize`	Returns the receiver's primary grouping size.

`init`	
`internationalCurrencySymbol`	Returns the international currency symbol used by the receiver.
`locale`	Returns the locale of the receiver.
`maximumFractionDigits`	Returns the maximum number of digits after the decimal separator allowed as input by the receiver.
`maximumIntegerDigits`	Returns the maximum number of integer digits allowed as input by the receiver.
`minimumFractionDigits`	Returns the minimum number of digits after the decimal separator allowed as input by the receiver.
`minimumIntegerDigits`	Returns the minimum number of integer digits allowed as input by the receiver.
`minusSign`	Returns the string the receiver uses to represent the minus sign.
`multiplier`	Returns the multiplier used by the receiver as an NSNumber object.
`negativeInfinitySymbol`	Returns the symbol the receiver uses to represent negative infinity.
`negativePrefix`	Returns the string the receiver inserts as a prefix to negative values.
`negativeSuffix`	Returns the string the receiver adds as a suffix to negative values.
`nilSymbol`	Returns the string the receiver uses to represent a nil value.
`notANumberSymbol`	Returns the symbol the receiver uses to represent NaN (“not a number”) when it converts values.
`numberFromString:`	Returns an NSNumber object created by parsing a given string.
`numberStyle`	Returns the number-formatter style of the receiver.
`paddingCharacter`	Returns a string containing the padding character for the receiver.
`paddingPosition`	Returns the padding position of the receiver.
`percentSymbol`	Returns the string that the receiver uses to represent the percent symbol.
`perMillSymbol`	Returns the string that the receiver uses for the per-thousands symbol.

`plusSign`	Returns the string the receiver uses for the plus sign.
`positiveInfinitySymbol`	Returns the string the receiver uses for the positive infinity symbol.
`positivePrefix`	Returns the string the receiver uses as the prefix for positive values.
`positiveSuffix`	Returns the string the receiver uses as the suffix for positive values.
`roundingIncrement`	Returns the rounding increment used by the receiver.
`roundingMode`	Returns the rounding mode used by the receiver.
`secondaryGroupingSize`	Returns the size of secondary groupings for the receiver.
`setAlwaysShowsDecimalSeparator:`	Controls whether the receiver always shows the decimal separator, even for integer numbers.
`setCurrencyCode:`	Sets the receiver's currency code.
`setCurrencyDecimalSeparator:`	Sets the string used by the receiver as a decimal separator.
`setCurrencySymbol:`	Sets the string used by the receiver as a local currency symbol.
`setExponentSymbol:`	Sets the string used by the receiver to represent the exponent symbol.
`setFormatterBehavior:`	Sets the formatter behavior of the receiver.
`setFormatWidth:`	Sets the format width used by the receiver.
`setGeneratesDecimalNumbers:`	Controls whether the receiver creates instances of NSDecimalNumber when it converts strings to number objects.
`setGroupingSeparator:`	Specifies the string used by the receiver for a grouping separator.
`setGroupingSize:`	Sets the grouping size of the receiver.
`setInternationalCurrencySymbol:`	Sets the string used by the receiver for the international currency symbol.
`setLocale:`	Sets the locale of the receiver.
`setMaximumFractionDigits:`	Sets the maximum number of digits after the decimal separator allowed as input by the receiver.
`setMaximumIntegerDigits:`	Sets the maximum number of integer digits allowed as input by the receiver.

`setMinimumFractionDigits:`	Sets the minimum number of digits after the decimal separator allowed as input by the receiver.
`setMinimumIntegerDigits:`	Sets the minimum number of integer digits allowed as input by the receiver.
`setMinusSign:`	Sets the string used by the receiver for the minus sign.
`setMultiplier:`	Sets the multiplier of the receiver.
`setNegativeInfinitySymbol:`	Sets the string used by the receiver for the negative infinity symbol.
`setNegativePrefix:`	Sets the string the receiver uses as a prefix for negative values.
`setNegativeSuffix:`	Sets the string the receiver uses as a suffix for negative values.
`setNilSymbol:`	Sets the string the receiver uses to represent nil values.
`setNotANumberSymbol:`	Sets the string the receiver uses to represent NaN (“not a number”).
`setNumberStyle:`	Sets the number style used by the receiver.
`setPaddingCharacter:`	Sets the string that the receiver uses to pad numbers in the formatted string representation.
`setPaddingPosition:`	Sets the padding position used by the receiver.
`setPercentSymbol:`	Sets the string used by the receiver to represent the percent symbol.
`setPerMillSymbol:`	Sets the string used by the receiver to represent the per-mill (per-thousand) symbol.
`setPlusSign:`	Sets the string used by the receiver to represent the plus sign.
`setPositiveInfinitySymbol:`	Sets the string used by the receiver for the positive infinity symbol.
`setPositivePrefix:`	Sets the string the receiver uses as the prefix for positive values.
`setPositiveSuffix:`	Sets the string the receiver uses as the suffix for positive values.
`setRoundingIncrement:`	Sets the rounding increment used by the receiver.
`setRoundingMode:`	Sets the rounding mode used by the receiver.
`setSecondaryGroupingSize:`	Sets the secondary grouping size of the receiver.

`setTextAttributesForNegativeInfinity:`	Sets the text attributes used to display the negative infinity symbol.
`setTextAttributesForNil:`	Sets the text attributes used to display the nil symbol.
`setTextAttributesForNotANumber:`	Sets the text attributes used to display the NaN ("not a number") string.
`setTextAttributesForPositiveInfinity:`	Sets the text attributes used to display the positive infinity symbol.
`setTextAttributesForZero:`	Sets the text attributes used to display a zero value.
`setUsesGroupingSeparator:`	Controls whether the receiver displays the grouping separator.
`setZeroSymbol:`	Sets the string the receiver uses as the symbol to show the value zero.
`stringFromNumber:`	Returns a string containing the formatted value of the provided number object.
`textAttributesForNegativeInfinity`	Returns a dictionary containing the text attributes used to display the negative infinity string.
`textAttributesForNil`	Returns a dictionary containing the text attributes used to display the nil symbol.
`textAttributesForNotANumber`	Returns a dictionary containing the text attributes used to display the NaN ("not a number") symbol.
`textAttributesForPositiveInfinity`	Returns a dictionary containing the text attributes used to display the positive infinity symbol.
`textAttributesForZero`	Returns a dictionary containing the text attributes used to display a value of zero.
`usesGroupingSeparator`	Returns a Boolean value that indicates whether the receiver uses the grouping separator.
`zeroSymbol`	Returns the string the receiver uses as the symbol to show the value zero.

NSObject

Complete reference information is available in the `NSObject` reference.

Class Methods

`classFallbacksForKeyedArchiver`	Overridden to return the names of classes that can be used to decode objects if their class is unavailable.

Instance Methods

`didChangeValueForKey:withSetMutation:usingObjects:`	Invoked to inform the receiver that the specified change was made to a specified unordered to-many relationship.
`finalize`	The garbage collector invokes this method on the receiver before disposing of the memory it uses.
`mutableSetValueForKey:`	Returns a mutable set proxy that provides read-write access to the unordered to-many relationship specified by a given key.
`mutableSetValueForKeyPath:`	Returns a mutable set that provides read-write access to the unordered to-many relationship specified by a given key path.
`willChangeValueForKey:withSetMutation: usingObjects:`	Invoked to inform the receiver that the specified change is about to be made to a specified unordered to-many relationship.

NSPredicate (New)

Complete reference information is available in the `NSPredicate` reference.

Class Methods

`predicateWithFormat:`	Creates and returns a new predicate formed by creating a new string with a given format and parsing the result.
`predicateWithFormat:argumentArray:`	Creates and returns a new predicate by substituting the values in a given array into a format string and parsing the result.
`predicateWithFormat:arguments:`	Creates and returns a new predicate by substituting the values in an argument list into a format string and parsing the result.
`predicateWithValue:`	Creates and returns a predicate that always evaluates to a given value.

Instance Methods

`evaluateWithObject:`	Returns a Boolean value that indicates whether a given object matches the conditions specified by the receiver.
`predicateFormat`	Returns the receiver's format string.
`predicateWithSubstitutionVariables:`	Returns a copy of the receiver with the receiver's variables substituted by values specified in a given substitution variables dictionary.

NSSet

Complete reference information is available in the `NSSet` reference.

Instance Methods

`addObserver:forKeyPath:options:context:`	Raises an exception.
`removeObserver:forKeyPath:`	Raises an exception.
`setValue:forKey:`	Invokes setValue:forKey: on each of the receiver's members.
`valueForKey:`	Return a set containing the results of invoking valueForKey: on each of the receiver's members.

NSString

Complete reference information is available in the `NSString` reference.

Class Methods

`stringWithContentsOfFile:encoding:error:`	Returns a string created by reading data from the file at a given path interpreted using a given encoding.
`stringWithContentsOfFile:usedEncoding:error:`	Returns a string created by reading data from the file at a given path and returns by reference the encoding used to interpret the file.
`stringWithContentsOfURL:encoding:error:`	Returns a string created by reading data from a given URL interpreted using a given encoding.

`stringWithContentsOfURL:usedEncoding:error:`	Returns a string created by reading data from a given URL and returns by reference the encoding used to interpret the data.
`stringWithCString:encoding:`	Returns a string containing the bytes in a given C array, interpreted according to a given encoding.

Instance Methods

`cStringUsingEncoding:`	Returns a representation of the receiver as a C string using a given encoding.
`getCString:maxLength:encoding:`	Converts the receiver's content to a given encoding and stores them in a buffer.
`initWithContentsOfFile:encoding:error:`	Returns an NSString object initialized by reading data from the file at a given path using a given encoding.
`initWithContentsOfFile:usedEncoding:error:`	Returns an NSString object initialized by reading data from the file at a given path and returns by reference the encoding used to interpret the characters.
`initWithContentsOfURL:encoding:error:`	Returns an NSString object initialized by reading data from a given URL interpreted using a given encoding.
`initWithContentsOfURL:usedEncoding:error:`	Returns an NSString object initialized by reading data from a given URL and returns by reference the encoding used to interpret the data.
`initWithCString:encoding:`	Returns an NSString object initialized using the characters in a given C array, interpreted according to a given encoding.
`lengthOfBytesUsingEncoding:`	Returns the number of bytes required to store the receiver in a given encoding.
`maximumLengthOfBytesUsingEncoding:`	Returns the maximum number of bytes needed to store the receiver in a given encoding.
`writeToFile:atomically:encoding:error:`	Writes the contents of the receiver to a file at a given path using a given encoding.
`writeToURL:atomically:encoding:error:`	Writes the contents of the receiver to the URL specified by url using the specified encoding.

NSURLDownload

Complete reference information is available in the `NSURLDownload` reference.

Class Methods

`canResumeDownloadDecodedWithEncodingMIMEType:`	Returns whether a URL download object can resume a download that was decoded with the specified MIME type.

Instance Methods

`deletesFileUponFailure`	Returns whether the receiver deletes partially downloaded files when a download stops prematurely.
`initWithResumeData:delegate:path:`	Returns an initialized NSURLDownload object that will resume downloading the specified data to the specified file and begins the download.
`resumeData`	Returns the resume data for a download that is not yet complete.
`setDeletesFileUponFailure:`	Specifies whether the receiver deletes the partially downloaded file when a download stops prematurely.

Delegate Methods

`download:willResumeWithResponse:fromByte:`	Sent when a download object has received a response from the server after attempting to resume a download.

NSURLRequest

Complete reference information is available in the `NSURLRequest` reference.

Instance Methods

`HTTPBodyStream`	Returns the receiver's HTTP body stream.

NSXMLDocument (New)

Complete reference information is available in the `NSXMLDocument` reference.

Class Methods

`replacementClassForClass:`	Overridden by subclasses to substitute a custom class for an NSXML class that the parser uses to create node instances.

Instance Methods

`addChild:`	Adds a child node after the last of the receiver's existing children.
`characterEncoding`	Returns the character encoding used for the XML.
`documentContentKind`	Returns the kind of document content for output.
`DTD`	Returns an NSXMLDTD object representing the internal DTD associated with the receiver.
`initWithContentsOfURL:options:error:`	Initializes and returns an NSXMLDocument object created from the XML or HTML contents of a URL-referenced source
`initWithData:options:error:`	Initializes and returns an NSXMLDocument object created from an NSData object.
`initWithRootElement:`	Returns an NSXMLDocument object initialized with a single child, the root element.
`initWithXMLString:options:error:`	Initializes and returns an NSXMLDocument object created from a string containing XML markup text.
`insertChild:atIndex:`	Inserts a node object at specified position in the receiver's array of children.
`insertChildren:atIndex:`	Inserts an array of children at a specified position in the receiver's array of children.
`isStandalone`	Returns whether the receiver represents a standalone XML document—that is, one without an external DTD.
`MIMEType`	Returns the MIME type for the receiver.
`objectByApplyingXSLT:arguments:error:`	Applies the XSLT pattern rules and templates (specified as a data object) to the receiver and returns a document object containing transformed XML or HTML markup.

`objectByApplyingXSLTAtURL:arguments:error:`	Applies the XSLT pattern rules and templates located at a specified URL to the receiver and returns a document object containing transformed XML markup or an NSData object containing plain text, RTF text, and so on.
`objectByApplyingXSLTString:arguments:error:`	Applies the XSLT pattern rules and templates (specified as a string) to the receiver and returns a document object containing transformed XML or HTML markup.
`removeChildAtIndex:`	Removes the child node of the receiver located at a specified position in its array of children.
`replaceChildAtIndex:withNode:`	Replaces the child node of the receiver located at a specified position in its array of children with another node.
`rootElement`	Returns the root element of the receiver.
`setCharacterEncoding:`	Sets the character encoding of the receiver to encoding,
`setChildren:`	Sets the child nodes of the receiver.
`setDocumentContentKind:`	Sets the kind of output content for the receiver.
`setDTD:`	Sets the internal DTD to be associated with the receiver.
`setMIMEType:`	Sets the MIME type of the receiver.
`setRootElement:`	Set the root element of the receiver.
`setStandalone:`	Sets a Boolean value that specifies whether the receiver represents a standalone XML document.
`setVersion:`	Sets the version of the receiver's XML.
`validateAndReturnError:`	Validates the document against the governing schema and returns whether the document conforms to the schema.
`version`	Returns the version of the receiver's XML.
`XMLData`	Returns the XML string representation of the receiver—that is, the entire document—encapsulated in a data object.
`XMLDataWithOptions:`	Returns the XML string representation of the receiver—that is, the entire document—encapsulated in a data object.

NSXMLDTD (New)

Complete reference information is available in the `NSXMLDTD` reference.

Class Methods

`predefinedEntityDeclarationForName:`	Returns a DTD node representing the predefined entity declaration with the specified name.

Instance Methods

`addChild:`	Adds a child node to the end of the list of existing children.
`attributeDeclarationForName:elementName:`	Returns the DTD node representing an attribute-list declaration for a given attribute and its element.
`elementDeclarationForName:`	Returns the DTD node representing an element declaration for a specified element.
`entityDeclarationForName:`	Returns the DTD node representing the entity declaration for a specified entity.
`initWithContentsOfURL:options:error:`	Initializes and returns an NSXMLDTD object created from the DTD declarations in a URL-referenced source.
`initWithData:options:error:`	Initializes and returns an NSXMLDTD object created from the DTD declarations encapsulated in an NSData object
`insertChild:atIndex:`	Inserts a child node in the receiver's list of children at a specific location in the list.
`insertChildren:atIndex:`	Inserts an array of child nodes at a specified location in the receiver's list of children.
`notationDeclarationForName:`	Returns the DTD node representing the notation declaration identified by the specified notation name.
`publicID`	Returns the receiver's public identifier.
`removeChildAtIndex:`	Removes the child node at a particular location in the receiver's list of children.
`replaceChildAtIndex:withNode:`	Replaces a child at a particular index with another child.
`setChildren:`	Removes all existing children of the receiver and replaces them with an array of new child nodes.

`setPublicID:`	Sets the public identifier of the receiver.
`setSystemID:`	Sets the system identifier of the receiver.
`systemID`	Returns the receiver's system identifier.

NSXMLDTDNode (New)

Complete reference information is available in the `NSXMLDTDNode` reference.

Instance Methods

`DTDKind`	Returns the receiver's DTD kind.
`initWithXMLString:`	Returns an NSXMLDTDNode object initialized with the DTD declaration in a given string.
`isExternal`	Returns a Boolean value that indicates whether the receiver represents a declaration from an external DTD (the system ID is set).
`notationName`	Returns the name of the notation associated with the receiver.
`publicID`	Returns the public identifier associated with the receiver.
`setDTDKind:`	Sets the receiver's DTD kind.
`setNotationName:`	Sets the notation name associated with the receiver.
`setPublicID:`	Sets the public identifier associated with the receiver.
`setSystemID:`	Sets the system identifier associated with the receiver.
`systemID`	Returns the system identifier associated with the receiver.

NSXMLElement (New)

Complete reference information is available in the `NSXMLElement` reference.

Instance Methods

`addAttribute:`	Adds an attribute node to the receiver.
`addChild:`	Adds a child node at the end of the receiver's current list of children.
`addNamespace:`	Adds a namespace node to the receiver.

`attributeForLocalName:URI:`	Returns the attribute node of the receiver that is identified by a local name and URI.
`attributeForName:`	Returns the attribute node of the receiver with the specified name.
`attributes`	Returns the receiver's attributes
`elementsForLocalName:URI:`	Returns the child element nodes (as NSXMLElement objects) of the receiver that are matched with the specified local name and URI.
`elementsForName:`	Returns the child element nodes (as NSXMLElement objects) of the receiver that have a specified name.
`initWithName:`	Returns an NSXMLElement object initialized with the specified name.
`initWithName:stringValue:`	Returns an NSXMLElement object initialized with a specified name and a single text-node child containing a specified value.
`initWithName:URI:`	Returns an NSXMLElement object initialized with the specified name and URI.
`initWithXMLString:error:`	Returns an NSXMLElement object created from a specified string containing XML markup.
`insertChild:atIndex:`	Inserts a new child node at a specified location in the receiver's list of child nodes.
`insertChildren:atIndex:`	Inserts an array of child nodes at a specified location in the receiver's list of children.
`namespaceForPrefix:`	Returns the namespace node with a specified prefix.
`namespaces`	Returns the namespace nodes of the receiver.
`normalizeAdjacentTextNodesPreservingCDATA:`	Coalesces adjacent text nodes of the receiver that you have explicitly added, optionally including CDATA sections.
`removeAttributeForName:`	Removes an attribute node that is identified by its name.
`removeChildAtIndex:`	Removes the child node of the receiver identified by a given index.
`removeNamespaceForPrefix:`	Removes a namespace node that is identified by a given prefix.
`replaceChildAtIndex:withNode:`	Replaces a child node at a specified location with another child node.

`resolveNamespaceForName:`	Returns the namespace node with the prefix matching the given qualified name.
`resolvePrefixForNamespaceURI:`	Returns the prefix associated with the specified URI.
`setAttributes:`	Sets all attributes of the receiver at once, replacing any existing attribute nodes.
`setAttributesAsDictionary:`	Sets the attributes of the receiver based on the key-value pairs specified in the passed-in dictionary.
`setChildren:`	Sets all child nodes of the receiver at once, replacing any existing children.
`setNamespaces:`	Sets all of the namespace nodes of the receiver at once, replacing any existing namespace nodes.

NSXMLNode (New)

Complete reference information is available in the `NSXMLNode` reference.

Class Methods

`attributeWithName:stringValue:`	Returns an NSXMLNode object representing an attribute node with a given name and string.
`attributeWithName:URI:stringValue:`	Returns an NSXMLNode object representing an attribute node with a given qualified name and string.
`commentWithStringValue:`	Returns an NSXMLNode object representing an comment node containing given text.
`document`	Returns an empty document node.
`documentWithRootElement:`	Returns an NSXMLDocument object initialized with a given root element.
`DTDNodeWithXMLString:`	Returns a NSXMLDTDNode object representing the DTD declaration for an element, attribute, entity, or notation based on a given string.
`elementWithName:`	Returns an NSXMLElement object with a given tag identifier, or name
`elementWithName:children:attributes:`	Returns an NSXMLElement object with the given tag (name), attributes, and children.

`elementWithName:stringValue:`	Returns an NSXMLElement object with a single text-node child containing the specified text.
`elementWithName:URI:`	Returns an element whose fully qualified name is specified.
`localNameForName:`	Returns the local name from the specified qualified name.
`namespaceWithName:stringValue:`	Returns an NSXMLNode object representing a namespace with a specified name and URI.
`predefinedNamespaceForPrefix:`	Returns an NSXMLNode object representing one of the predefined namespaces with the specified prefix.
`prefixForName:`	Returns the prefix from the specified qualified name.
`processingInstructionWithName:stringValue:`	Returns an NSXMLNode object representing a processing instruction with a specified name and value.
`textWithStringValue:`	Returns an NSXMLNode object representing a text node with specified content.

Instance Methods

`canonicalXMLStringPreservingComments:`	Returns a string object encapsulating the receiver's XML in canonical form.
`childAtIndex:`	Returns the child node of the receiver at the specified location.
`childCount`	Returns the number of child nodes the receiver has.
`children`	Returns an immutable array containing the child nodes of the receiver (as NSXMLNode objects).
`description`	Returns a description of the receiver.
`detach`	Detaches the receiver from its parent node.
`index`	Returns the index of the receiver identifying its position relative to its sibling nodes.
`initWithKind:`	Returns an NSXMLNode instance initialized with the constant indicating node kind.
`initWithKind:options:`	Returns an NSXMLNode instance initialized with the constant indicating node kind and one or more initialization options.

`kind`	Returns the kind of node the receiver is as a constant of type NSXMLNodeKind.
`level`	Returns the nesting level of the receiver within the tree hierarchy.
`localName`	Returns the local name of the receiver.
`name`	Returns the name of the receiver.
`nextNode`	Returns the next NSXMLNode object in document order.
`nextSibling`	Returns the next NSXMLNode object that is a sibling node to the receiver.
`nodesForXPath:error:`	Returns the nodes resulting from executing an XPath query upon the receiver.
`objectsForXQuery:constants:error:`	Returns the objects resulting from executing an XQuery query upon the receiver.
`objectsForXQuery:error:`	Returns the objects resulting from executing an XQuery query upon the receiver.
`objectValue`	Returns the object value of the receiver.
`parent`	Returns the parent node of the receiver.
`prefix`	Returns the prefix of the receiver's name.
`previousNode`	Returns the previous NSXMLNode object in document order.
`previousSibling`	Returns the previous NSXMLNode object that is a sibling node to the receiver.
`rootDocument`	Returns the NSXMLDocument object containing the root element and representing the XML document as a whole.
`setName:`	Sets the name of the receiver.
`setObjectValue:`	Sets the content of the receiver to an object value.
`setStringValue:`	Sets the content of the receiver as a string value.
`setStringValue:resolvingEntities:`	Sets the content of the receiver as a string value and, optionally, resolves character references, predefined entities, and user-defined entities as declared in the associated DTD.
`setURI:`	Sets the URI of the receiver.
`stringValue`	Returns the content of the receiver as a string value.

`URI`	Returns the URI associated with the receiver.
`XMLString`	Returns the string representation of the receiver as it would appear in an XML document.
`XMLStringWithOptions:`	Returns the string representation of the receiver as it would appear in an XML document, with one or more output options specified.
`XPath`	Returns the XPath expression identifying the receiver's location in the document tree.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

FoundationErrors.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFileErrorMaximum`	Marks the end of the range of error codes reserved for file errors
`NSFileErrorMinimum`	Marks the start of the range of error codes reserved for file errors
`NSFileLockingError`	Failure to get a lock on file
`NSFileNoSuchFileError`	File-system operation attempted on non-existent file
`NSFileReadCorruptFileError`	Read error because of a corrupted file, bad format, or similar reason
`NSFileReadInapplicableStringEncodingError`	Read error because the string encoding was not applicable.
`NSFileReadInvalidFileNameError`	Read error because of an invalid file name
`NSFileReadNoPermissionError`	Read error because of a permission problem
`NSFileReadNoSuchFileError`	Read error because no such file was found
`NSFileReadUnknownError`	Read error, reason unknown

NSFileReadUnsupportedSchemeError	Read error because the specified URL scheme is unsupported
NSFileWriteInapplicableStringEncodingError	Write error because the string encoding was not applicable.
NSFileWriteInvalidFileNameError	Write error because of an invalid file name
NSFileWriteNoPermissionError	Write error because of a permission problem
NSFileWriteOutOfSpaceError	Write error because of a lack of disk space
NSFileWriteUnknownError	Write error, reason unknown
NSFileWriteUnsupportedSchemeError	Write error because the specified URL scheme is unsupported
NSFormattingError	Formatting error (related to display of data)
NSFormattingErrorMaximum	Marks end of the range of error codes reserved for formatting errors.
NSFormattingErrorMinimum	Marks the start of the range of error codes reserved for formatting errors.
NSKeyValueValidationError	Key-value coding validation error
NSUserCancelledError	The user cancelled the operation (for example, by pressing Command-period).
NSValidationErrorMaximum	Marks the start and end of the range of error codes reserved for validation errors.
NSValidationErrorMinimum	Marks the start of the range of error codes reserved for validation errors.

NSAffineTransform.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSAffineTransformStruct	This type defines the three-by-three matrix that performs an affine transform between two coordinate systems.

NSCalendar.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSCalendarUnit`	Defines a type used to specify calendrical units such as day and month.
`NSDayCalendarUnit`	Specifies the day unit.
`NSEraCalendarUnit`	Specifies the era unit.
`NSHourCalendarUnit`	Specifies the hour unit.
`NSMinuteCalendarUnit`	Specifies the minute unit.
`NSMonthCalendarUnit`	Specifies the month unit.
`NSSecondCalendarUnit`	Specifies the second unit.
`NSUndefinedDateComponent`	Specifies that the component is undefined.
`NSWeekCalendarUnit`	Specifies the week unit.
`NSWeekdayCalendarUnit`	Specifies the weekday unit.
`NSWeekdayOrdinalCalendarUnit`	Specifies the ordinal weekday unit.
`NSWrapCalendarComponents`	Specifies that the components specified for an NSDateComponents object should be incremented and wrap around to zero/one on overflow, but should not cause higher units to be incremented.
`NSYearCalendarUnit`	Specifies the year unit.

NSComparisonPredicate.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSAllPredicateModifier`	A predicate to compare all entries in the destination of a to-many relationship.
`NSAnyPredicateModifier`	A predicate to match with any entry in the destination of a to-many relationship.

NSBeginsWithPredicateOperatorType	A begins-with predicate.
NSCaseInsensitivePredicateOption	A case-insensitive predicate.
NSComparisonPredicateModifier	These constants describe the possible types of modifier for NSComparisonPredicate.
NSCustomSelectorPredicateOperatorType	Predicate that uses a custom selector that takes a single argument and returns a BOOL value.
NSDiacriticInsensitivePredicateOption	A diacritic-insensitive predicate.
NSDirectPredicateModifier	A predicate to compare directly the left and right hand sides.
NSEndsWithPredicateOperatorType	An ends-with predicate.
NSEqualToPredicateOperatorType	An equal-to predicate.
NSGreaterThanOrEqualToPredicateOperatorType	A greater-than-or-equal-to predicate.
NSGreaterThanPredicateOperatorType	A greater-than predicate.
NSInPredicateOperatorType	A predicate to determine if the left hand side is in the right hand side.
NSLessThanOrEqualToPredicateOperatorType	A less-than-or-equal-to predicate.
NSLessThanPredicateOperatorType	A less-than predicate.
NSLikePredicateOperatorType	A simple subset of the matches predicate, similar in behavior to SQL LIKE.
NSMatchesPredicateOperatorType	A full regular expression matching predicate.
NSNotEqualToPredicateOperatorType	A not-equal-to predicate.
NSPredicateOperatorType	Defines the type of comparison for NSComparisonPredicate.

NSCompoundPredicate.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSAndPredicateType	A logical AND predicate.
NSCompoundPredicateType	These constants describe the possible types of NSCompoundPredicate.

`NSNotPredicateType`	A logical NOT predicate.
`NSOrPredicateType`	A logical OR predicate.

NSData.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSAtomicWrite`	A hint to use an auxiliary file when saving data and then exchange the files.
`NSMappedRead`	A hint indicating the file should be mapped into virtual memory, if possible.
`NSUncachedRead`	A hint indicating the file should not be stored in the file-system caches.

NSDateFormatter.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSDateFormatterBehavior`	Constants that specify the behavior NSDateFormatter should exhibit.
`NSDateFormatterBehavior10_0`	Specifies formatting behavior equivalent to that in Mac OS X 10.0.
`NSDateFormatterBehavior10_4`	Specifies formatting behavior equivalent for Mac OS X 10.4.
`NSDateFormatterBehaviorDefault`	Specifies default formatting behavior.
`NSDateFormatterFullStyle`	Specifies a full style with complete details, such as “Tuesday, April 12, 1952 AD” or “3:30:42pm PST”.
`NSDateFormatterLongStyle`	Specifies a long style, typically with full text, such as “November 23, 1937” or “3:30:32pm”.
`NSDateFormatterMediumStyle`	Specifies a medium style, typically with abbreviated text, such as “Nov 23, 1937”.
`NSDateFormatterNoStyle`	Specifies no style.

NSDateFormatterShortStyle	Specifies a short style, typically numeric only, such as “11/23/37” or “3:30pm”.
NSDateFormatterStyle	The following constants specify predefined date and time format styles.

NSError.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSCocoaErrorDomain	Application Kit and Foundation Kit errors.
NSFilePathErrorKey	Contains the file path of the error.
NSLocalizedFailureReasonErrorKey	The corresponding value is a localized string representation containing the reason for the failure that, if present, will be returned by localizedFailureReason.
NSLocalizedRecoveryOptionsErrorKey	The corresponding value is an array containing the localized titles of buttons appropriate for displaying in an alert panel.
NSLocalizedRecoverySuggestionErrorKey	The corresponding value is a string containing the localized recovery suggestion for the error.
NSRecoveryAttempterErrorKey	The corresponding value is an object that conforms to the NSErrorRecoveryAttempting informal protocol.
NSStringEncodingErrorKey	The corresponding value is an NSNumber object containing the NSStringEncoding value.
NSURLErrorKey	The corresponding value is an NSURL object.

NSExpression.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSConstantValueExpressionType	An expression that always returns the same value.
NSEvaluatedObjectExpressionType	An expression that always returns the parameter object itself.
NSExpressionType	Defines the possible types of NSExpression.

NSFunctionExpressionType	An expression that returns the result of evaluating a function.
NSKeyPathExpressionType	An expression that returns something that can be used as a key path.
NSVariableExpressionType	An expression that always returns whatever value is associated with the key specified by ‘variable’ in the bindings dictionary.

NSFileManager.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSFileBusy	The key in a file attribute dictionary whose value indicates whether the file is busy.

NSKeyValueCoding.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSAverageKeyValueOperator	The @avg array operator.
NSCountKeyValueOperator	The @count array operator.
NSDistinctUnionOfArraysKeyValueOperator	The @distinctUnionOfArrays array operator.
NSDistinctUnionOfObjectsKeyValueOperator	The @distinctUnionOfObjects array operator.
NSDistinctUnionOfSetsKeyValueOperator	The @distinctUnionOfSets array operator.
NSMaximumKeyValueOperator	The @max array operator.
NSMinimumKeyValueOperator	The @min array operator.
NSSumKeyValueOperator	The @sum array operator.
NSUnionOfArraysKeyValueOperator	The @unionOfArrays array operator.
NSUnionOfObjectsKeyValueOperator	The @unionOfObjects array operator.
NSUnionOfSetsKeyValueOperator	The @unionOfSets array operator.

NSKeyValueObserving.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSKeyValueIntersectSetMutation`	Indicates that the objects not in the specified set are being removed from the receiver. This mutation kind results in a NSkeyValueChangeKindKey value of NSKeyValueChangeRemoval.
`NSKeyValueMinusSetMutation`	Indicates that the objects in the specified set are being removed from the receiver. This mutation kind results in a NSkeyValueChangeKindKey value of NSKeyValueChangeRemoval.
`NSKeyValueSetMutationKind`	These constants are specified as the parameter to the methods willChangeValueForKey: withSetMutation:usingObjects: and didChangeValueForKey: withSetMutation:usingObjects: .
`NSKeyValueSetSetMutation`	Indicates that set of objects are replacing the existing objects in the receiver. This mutation kind results in a NSkeyValueChangeKindKey value of NSKeyValueChangeReplacement.
`NSKeyValueUnionSetMutation`	Indicates that objects in the specified set are being added to the receiver. This mutation kind results in a NSkeyValueChangeKindKey value of NSKeyValueChangeInsertion.

NSLocale.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSBuddhistCalendar`	Identifier for the Buddhist calendar.
`NSChineseCalendar`	Identifier for the Chinese calendar (unsupported).
`NSGregorianCalendar`	Identifier for the Gregorian calendar.
`NSHebrewCalendar`	Identifier for the Hebrew calendar.
`NSIslamicCalendar`	Identifier for the Islamic calendar.
`NSIslamicCivilCalendar`	Identifier for the Islamic civil calendar.
`NSJapaneseCalendar`	Identifier for the Japanese calendar.
`NSLocaleCalendar`	The key for the calendar associated with the locale.

NSLocaleCollationIdentifier	The key for the collation associated with the locale.
NSLocaleCountryCode	The key for the locale country code.
NSLocaleCurrencyCode	The key for the currency code associated with the locale.
NSLocaleCurrencySymbol	The key for the currency symbol associated with the locale.
NSLocaleDecimalSeparator	The key for the decimal separator associated with the locale.
NSLocaleExemplarCharacterSet	The key for the exemplar character set for the locale.
NSLocaleGroupingSeparator	The key for the numeric grouping separator associated with the locale.
NSLocaleIdentifier	The key for the locale identifier.
NSLocaleLanguageCode	The key for the locale language code.
NSLocaleMeasurementSystem	The key for the measurement system associated with the locale.
NSLocaleScriptCode	The key for the locale script code.
NSLocaleUsesMetricSystem	The key for the flag that indicates whether the locale uses the metric system.
NSLocaleVariantCode	The key for the locale variant code.

NSMetadata.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSMetadataQueryDidFinishGatheringNotification	Posted when the receiver has finished with the initial result-gathering phase of the query.
NSMetadataQueryDidStartGatheringNotification	Posted when the receiver begins with the initial result-gathering phase of the query.
NSMetadataQueryDidUpdateNotification	Posted when the receiver's results have changed during the live-update phase of the query.
NSMetadataQueryGatheringProgressNotification	Posted as the receiver's is collecting results during the initial result-gathering phase of the query.

NSMetadataQueryLocalComputerScope	Search all local mounted volumes, including the user home directory. The user's home directory is searched even if it is a remote volume.
NSMetadataQueryNetworkScope	Search all user-mounted remote volumes.
NSMetadataQueryResultContentRelevanceAttribute	Key used to retrieve an NSNumber object with a floating point value between 0.0 and 1.0 inclusive.
NSMetadataQueryUserHomeScope	Search the user's home directory.
NSMQA1	

NSNetServices.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSNetServicesTimeoutError	The net service has timed out.

NSNumberFormatter.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSNumberFormatterBehavior	These constants specify the behavior of a number formatter.
NSNumberFormatterBehavior10_0	The number-formatter behavior as it existed prior to Mac OS X v10.4.
NSNumberFormatterBehavior10_4	The number-formatter behavior since Mac OS X v10.4.
NSNumberFormatterBehaviorDefault	The number-formatter behavior set as the default for new instances. You can set the default formatter behavior with the class method setDefaultFormatterBehavior:.
NSNumberFormatterCurrencyStyle	Specifies a currency style format.
NSNumberFormatterDecimalStyle	Specifies a decimal style format.
NSNumberFormatterNoStyle	Specifies no style.

NSNumberFormatterPadAfterPrefix	Specifies that the padding should occur after the prefix.
NSNumberFormatterPadAfterSuffix	Specifies that the padding should occur after the suffix.
NSNumberFormatterPadBeforePrefix	Specifies that the padding should occur before the prefix.
NSNumberFormatterPadBeforeSuffix	Specifies that the padding should occur before the suffix.
NSNumberFormatterPadPosition	These constants are used to specify how numbers should be padded.
NSNumberFormatterPercentStyle	Specifies a percent style format.
NSNumberFormatterRoundCeiling	Round up to next larger number with the proper number of digits after the decimal separator.
NSNumberFormatterRoundDown	Round down to next smaller number with the proper number of digits after the decimal separator.
NSNumberFormatterRoundFloor	Round down to next smaller number with the proper number of digits after the decimal separator.
NSNumberFormatterRoundHalfDown	Round down when a 5 follows putative last digit.
NSNumberFormatterRoundHalfEven	Round the last digit, when followed by a 5, toward an even digit (.25 -> .2, .35 -> .4)
NSNumberFormatterRoundHalfUp	Round up when a 5 follows putative last digit.
NSNumberFormatterRoundingMode	These constants are used to specify how numbers should be rounded.
NSNumberFormatterRoundUp	Round up to next larger number with the proper number of digits after the decimal separator.
NSNumberFormatterScientificStyle	Specifies a scientific style format.
NSNumberFormatterSpellOutStyle	Specifies a spell-out format; for example, “23” becomes “twenty-three”.
NSNumberFormatterStyle	These constants specify predefined number format styles.

NSObjCRuntime.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSFoundationVersionNumber10_3	Foundation version released in Mac OS X version 10.3.
NSFoundationVersionNumber10_3_2	Foundation version released in Mac OS X version 10.3.2.

`NSFoundationVersionNumber10_3_3`	Foundation version released in Mac OS X version 10.3.3.
`NSFoundationVersionNumber10_3_4`	Foundation version released in Mac OS X version 10.3.4.

NSPathUtilities.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSApplicationSupportDirectory`	Location of application support files (Library/Application Support).
`NSCachesDirectory`	Location of discardable cache files (Library/Caches).
`NSCoreServiceDirectory`	Location of core services (System/Library/CoreServices).
`NSDesktopDirectory`	Location of user's desktop directory.

NSURLError.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLErrorClientCertificateRejected`	Returned when a server certificate is rejected.
`NSURLErrorServerCertificateNotYetValid`	Returned when a server certificate is not yet valid.

NSXMLDTDNode.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSXMLAttributeCDATAKind`	Identifies an attribute-list declaration with a CDATA (character data) value type.
`NSXMLAttributeEntitiesKind`	Identifies an attribute-list declaration with an ENTITIES value type (refers to multiple unparsed entities declared elsewhere in document).

`NSXMLAttributeEntityKind`	Identifies an attribute-list declaration with an ENTITY value type (refers to unparsed entity declared in document).
`NSXMLAttributeEnumerationKind`	Identifies an attribute-list declaration with an enumeration value type (list of all possible values).
`NSXMLAttributeIDKind`	Identifies an attribute-list declaration with an ID value type (per-document unique element name).
`NSXMLAttributeIDRefKind`	Identifies an attribute-list declaration with an IDREF value type (refers to element ID type).
`NSXMLAttributeIDRefsKind`	Identifies an attribute-list declaration with an IDREFS value type (refers to multiple elements of ID type).
`NSXMLAttributeNMTokenKind`	Identifies an attribute-list declaration with a NMTOKEN value type (name token).
`NSXMLAttributeNMTokensKind`	Identifies an attribute-list declaration with a NMTOKENS value type (multiple name tokens)
`NSXMLAttributeNotationKind`	Identifies an attribute-list declaration with a NOTATION value type (name of declared notation).
`NSXMLDTDNodeKind`	The type defined for the constants that specify the kind and subkind of DTD declaration represented by an NSXMLDTDNode object. You set the DTD-node kind using the setDTDKind: method.
`NSXMLElementDeclarationAnyKind`	Identifies an ANY element declaration.
`NSXMLElementDeclarationElementKind`	Identifies a declaration of an element with child elements.
`NSXMLElementDeclarationEmptyKind`	Identifies a declaration (EMPTY) of an empty element.
`NSXMLElementDeclarationMixedKind`	Identifies a declaration of an element with mixed content ((#PCDATA | child)).
`NSXMLElementDeclarationUndefinedKind`	Identifies an undefined element declaration.
`NSXMLEntityGeneralKind`	Identifies a general entity declaration.
`NSXMLEntityParameterKind`	Identifies a parameter entity declaration.
`NSXMLEntityParsedKind`	Identifies a parsed entity declaration.
`NSXMLEntityPredefined`	Identifies a predefined entity declaration.
`NSXMLEntityUnparsedKind`	Identifies an unparsed entity declaration.

NSXMLDocument.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSXMLDocumentContentKind`	Type used to define the kind of document content.
`NSXMLDocumentHTMLKind`	Outputs empty tags in HTML without a close tag, such as .
`NSXMLDocumentTextKind`	Outputs the string value of the document by extracting the string values from all text nodes.
`NSXMLDocumentXHTMLKind`	The document output is XHTML.
`NSXMLDocumentXMLKind`	The default type of document content type, which is XML.

NSXMLNode.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSXMLAttributeDeclarationKind`	Specifies an attribute-list declaration node.
`NSXMLAttributeKind`	Specifies an attribute node
`NSXMLCommentKind`	Specifies a comment node.
`NSXMLDocumentKind`	Specifies a document node.
`NSXMLDTDKind`	Specifies a document-type declaration (DTD) node.
`NSXMLElementDeclarationKind`	Specifies an element declaration node.
`NSXMLElementKind`	Specifies an element node.
`NSXMLEntityDeclarationKind`	Specifies an entity-declaration node.
`NSXMLInvalidKind`	Indicates a node object created without a valid kind being specified (as returned by the kind method).
`NSXMLNamespaceKind`	Specifies a namespace node.
`NSXMLNodeKind`	A type defined for the node-kind constants described in “Node Kind Constants.”
`NSXMLNotationDeclarationKind`	Specifies a notation declaration node.

`NSXMLProcessingInstructionKind`	Specifies a processing-instruction node.
`NSXMLTextKind`	Specifies a text node.

NSXMLNodeOptions.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSXMLDocumentIncludeContentTypeDeclaration`	Includes a content type declaration for HTML or XHTML in the output of the document.
`NSXMLDocumentTidyHTML`	Formats HTML into valid XHTML during processing of the document.
`NSXMLDocumentTidyXML`	Changes malformed XML into valid XML during processing of the document.
`NSXMLDocumentValidate`	Validates this document against its DTD (internal or external) or XML Schema.
`NSXMLDocumentXInclude`	Replaces all XInclude nodes in the document with the nodes referred to.
`NSXMLNodeCompactEmptyElement`	Requests that an element should be contracted when empty; for example, <flag/>.
`NSXMLNodeExpandEmptyElement`	Requests that an element should be expanded when empty; for example, <flag></flag>. This is the default.
`NSXMLNodeIsCDATA`	Specifies that a text node contains and is written out as a CDATA section.
`NSXMLNodeOptionsNone`	No options are requested for this input or output action.
`NSXMLNodePreserveAll`	Turns on all preservation options: attribute and namespace order, entities, prefixes, CDATA, whitespace, quotes, and empty elements.
`NSXMLNodePreserveAttributeOrder`	Requests that NSXMLNode preserve the order of attributes as in the source XML.
`NSXMLNodePreserveCDATA`	Requests that NSXMLNode preserve CDATA blocks where defined in the input XML.

`NSXMLNodePreserveCharacterReferences`	Specifies that character references (&#nnn;) should not be resolved for XML output of this node.
`NSXMLNodePreserveDTD`	Specifies that declarations in a DTD should be preserved until it the DTD is modified. For example, parameter entities are by default expanded; with this option, they are written out as they originally occur in the DTD.
`NSXMLNodePreserveEmptyElements`	Specifies that empty elements in the input XML be preserved in their contracted or expanded form.
`NSXMLNodePreserveEntities`	Specifies that entities (&xyz;) should not be resolved for XML output of this node.
`NSXMLNodePreserveNamespaceOrder`	Requests NSXML to preserve the order of namespace URI definitions as in the source XML.
`NSXMLNodePreservePrefixes`	Requests NSXMLNode not to choose prefixes based on the closest namespace URI definition.
`NSXMLNodePreserveQuotes`	Specifies that the quoting style used in the input XML (single or double quotes) be preserved.
`NSXMLNodePreserveWhitespace`	Requests NSXMLNode to preserve whitespace characters (such as tabs and carriage returns) in the XML source that are not part of node content.
`NSXMLNodePrettyPrint`	Print this node with extra space for readability. (Output)
`NSXMLNodeUseDoubleQuotes`	Requests that NSXML use double quotes for the value of an attribute or namespace node. This is the default.
`NSXMLNodeUseSingleQuotes`	Requests that NSXML use single quotes for the value of an attribute or namespace node.

NSZone.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSAllocateCollectable`	Allocates collectable memory.
`NSReallocateCollectable`	Reallocates collectable memory.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSScannedOption`	Specifies allocation of scanned memory.

10.3 Symbol Changes

This article lists the symbols added to `Foundation.framework` in Mac OS X v10.3.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

NSAppleEventManager

Complete reference information is available in the `NSAppleEventManager` reference.

Instance Methods

`appleEventForSuspensionID:`	Given a nonzero suspensionID returned by an invocation of suspendCurrentAppleEvent, returns the descriptor for the event whose handling was suspended.
`currentAppleEvent`	Returns the descriptor for currentAppleEvent if an Apple event is being handled on the current thread.
`currentReplyAppleEvent`	Returns the corresponding reply event descriptor if an Apple event is being handled on the current thread.
`replyAppleEventForSuspensionID:`	Given a nonzero suspensionID returned by an invocation of suspendCurrentAppleEvent, returns the corresponding reply event descriptor.
`resumeWithSuspensionID:`	Given a nonzero suspensionID returned by an invocation of suspendCurrentAppleEvent, signal that handling of the suspended event may now continue.

`setCurrentAppleEventAndReplyEventWithSuspensionID:`	Given a nonzero suspensionID returned by an invocation of suspendCurrentAppleEvent, sets the values that will be returned by subsequent invocations of currentAppleEvent and currentReplyAppleEvent to be the event whose handling was suspended and its corresponding reply event, respectively.
`suspendCurrentAppleEvent`	Suspends the handling of the current event and returns an ID that must be used to resume the handling of the event if an Apple event is being handled on the current thread.

NSArray

Complete reference information is available in the `NSArray` reference.

Instance Methods

`addObserver:toObjectsAtIndexes: forKeyPath:options:context:`	Registers anObserver to receive key value observer notifications for the specified keypath relative to the objects at indexes.
`removeObserver:fromObjectsAtIndexes:forKeyPath:`	Removes anObserver from all key value observer notifications associated with the specified keyPath relative to the receiver's objects at indexes.
`setValue:forKey:`	Invokes setValue:forKey: on each of the receiver's items using the specified value and key.
`sortedArrayUsingDescriptors:`	Returns a copy of the receiver sorted as specified by a given array of sort descriptors.
`valueForKey:`	Returns an array containing the results of invoking valueForKey: using key on each of the receiver's objects.

NSCharacterSet

Complete reference information is available in the `NSCharacterSet` reference.

Class Methods

`symbolCharacterSet`	Returns a character set containing the characters in the category of Symbols.

NSDictionary

Complete reference information is available in the `NSDictionary` reference.

Instance Methods

`valueForKey:`	Returns the value associated with a given key.

NSDistributedNotificationCenter

Complete reference information is available in the `NSDistributedNotificationCenter` reference.

Instance Methods

`postNotificationName:object:userInfo:options:`	Creates a notification with information, and posts it to the receiver.

NSIndexSet (New)

Complete reference information is available in the `NSIndexSet` reference.

Class Methods

`indexSet`	Creates an empty index set.
`indexSetWithIndex:`	Creates an index set with an index.
`indexSetWithIndexesInRange:`	Creates an index set with an index range.

Instance Methods

`containsIndex:`	Indicates whether the receiver contains a specific index.
`containsIndexes:`	Indicates whether the receiver contains a superset of the indexes in another index set.

`containsIndexesInRange:`	Indicates whether the receiver contains the indexes represented by an index range.
`count`	Returns the number of indexes in the receiver.
`firstIndex`	Returns either the first index in the receiver or the not-found indicator.
`getIndexes:maxCount:inIndexRange:`	The receiver fills an index buffer with the indexes contained both in the receiver and in an index range, returning the number of indexes copied.
`indexGreaterThanIndex:`	Returns either the closest index in the receiver that is greater than a specific index or the not-found indicator.
`indexGreaterThanOrEqualToIndex:`	Returns either the closest index in the receiver that is greater than or equal to a specific index or the not-found indicator.
`indexLessThanIndex:`	Returns either the closest index in the receiver that is less than a specific index or the not-found indicator.
`indexLessThanOrEqualToIndex:`	Returns either the closest index in the receiver that is less than or equal to a specific index or the not-found indicator.
`init`	Initializes an allocated NSIndexSet object.
`initWithIndex:`	Initializes an allocated NSIndexSet object with an index.
`initWithIndexesInRange:`	Initializes an allocated NSIndexSet object with an index range.
`initWithIndexSet:`	Initializes an allocated NSIndexSet object with an index set.
`intersectsIndexesInRange:`	Indicates whether the receiver contains any of the indexes in a range.
`isEqualToIndexSet:`	Indicates whether the indexes in the receiver are the same indeces contained in another index set.
`lastIndex`	Returns either the last index in the receiver or the not-found indicator.

NSInputStream (New)

Complete reference information is available in the `NSInputStream` reference.

Class Methods

`inputStreamWithData:`	Creates and returns an initialized NSInputStream object for reading from a given NSData object.

`inputStreamWithFileAtPath:`	Creates and returns an initialized NSInputStream object that reads data from the file at a given path.

Instance Methods

`getBuffer:length:`	Returns by reference a pointer to a read buffer and, by reference, the number of bytes available, and returns a Boolean value that indicates whether the buffer is available.
`hasBytesAvailable`	Returns a Boolean value that indicates whether the receiver has bytes available to read.
`initWithData:`	Initializes and returns an NSInputStream object for reading from a given NSData object.
`initWithFileAtPath:`	Initializes and returns an NSInputStream object that reads data from the file at a given path.
`read:maxLength:`	Reads up to a given number of bytes into a given buffer, and returns the actual number of bytes read.

NSMutableArray

Complete reference information is available in the `NSMutableArray` reference.

Instance Methods

`sortUsingDescriptors:`	Sorts the receiver using a given array of sort descriptors.

NSMutableDictionary

Complete reference information is available in the `NSMutableDictionary` reference.

Instance Methods

`setValue:forKey:`	Adds a given key-value pair to the receiver.

NSMutableIndexSet (New)

Complete reference information is available in the `NSMutableIndexSet` reference.

Instance Methods

`addIndex:`	Adds an index to the receiver.
`addIndexes:`	Adds the indexes in an index set to the receiver.
`addIndexesInRange:`	Adds the indexes in an index range to the receiver.
`removeAllIndexes`	Removes the receiver's indexes.
`removeIndex:`	Removes an index from the receiver.
`removeIndexes:`	Removes the indexes in an index set from the receiver.
`removeIndexesInRange:`	Removes the indexes in an index range from the receiver.
`shiftIndexesStartingAtIndex:by:`	Shifts a group of indexes to the left or the right within the receiver.

NSObject

Complete reference information is available in the `NSObject` reference.

Class Methods

`automaticallyNotifiesObserversForKey:`	Returns a Boolean value that indicates whether the receiver supports automatic key-value observation for the given key.
`setKeys:triggerChangeNotificationsForDependentKey:`	Configures the receiver to post change notifications for a given property if any of the properties specified in a given array changes.

Instance Methods

`addObserver:forKeyPath:options:context:`	Registers anObserver to receive KVO notifications for the specified key-path relative to the receiver.
`dictionaryWithValuesForKeys:`	Returns a dictionary containing the property values identified by each of the keys in a given array.
`didChange:valuesAtIndexes:forKey:`	Invoked to inform the receiver that the specified change has occurred on the indexes for a specified ordered to-many relationship.

`didChangeValueForKey:`	Invoked to inform the receiver that the value of a given property has changed.
`mutableArrayValueForKey:`	Returns a mutable array proxy that provides read-write access to an ordered to-many relationship specified by a given key.
`mutableArrayValueForKeyPath:`	Returns a mutable array that provides read-write access to the ordered to-many relationship specified by a given key path.
`observationInfo`	Returns a pointer that identifies information about all of the observers that are registered with the receiver.
`observeValueForKeyPath:ofObject:change:context:`	This message is sent to the receiver when the value at the specified key path relative to the given object has changed.
`removeObserver:forKeyPath:`	Stops a given object from receiving change notifications for the property specified by a given key-path relative to the receiver.
`setNilValueForKey:`	Invoked by setValue:forKey: when it's given a nil value for a scalar value (such as an int or float).
`setObservationInfo:`	Sets the observation info for the receiver.
`setValue:forKey:`	Sets the property of the receiver specified by a given key to a given value.
`setValue:forKeyPath:`	Sets the value for the property identified by a given key path to a given value.
`setValue:forUndefinedKey:`	Invoked by setValue:forKey: when it finds no property for a given key.
`setValuesForKeysWithDictionary:`	Sets properties of the receiver with values from a given dictionary, using its keys to identify the properties.
`validateValue:forKey:error:`	Returns a Boolean value that indicates whether the value specified by a given pointer is valid for the property identified by a given key.
`validateValue:forKeyPath:error:`	Returns a Boolean value that indicates whether the value specified by a given pointer is valid for a given key path relative to the receiver.

`valueForUndefinedKey:`	Invoked by valueForKey: when it finds no property corresponding to a given key.
`willChange:valuesAtIndexes:forKey:`	Invoked to inform the receiver that the specified change is about to be executed at given indexes for a specified ordered to-many relationship.
`willChangeValueForKey:`	Invoked to inform the receiver that the value of a given property is about to change.

NSOutputStream (New)

Complete reference information is available in the `NSOutputStream` reference.

Class Methods

`outputStreamToBuffer:capacity:`	Creates and returns an initialized output stream that can write to a provided buffer.
`outputStreamToFileAtPath:append:`	Creates and returns an initialized output stream for writing to a specified file.
`outputStreamToMemory`	Creates and returns an initialized output stream that will write stream data to memory.

Instance Methods

`hasSpaceAvailable`	Returns whether the receiver can be written to.
`initToBuffer:capacity:`	Returns an initialized output stream that can write to a provided buffer.
`initToFileAtPath:append:`	Returns an initialized output stream for writing to a specified file.
`initToMemory`	Returns an initialized output stream that will write to memory.
`write:maxLength:`	Writes the contents of a provided data buffer to the receiver.

NSScriptCommand

Complete reference information is available in the `NSScriptCommand` reference.

Class Methods

`currentCommand`	If a command is being executed in the current thread by Cocoa scripting's built-in Apple event handling, return the command.

Instance Methods

`appleEvent`	If the receiver was constructed by Cocoa scripting's built-in Apple event handling, returns the Apple event descriptor from which it was constructed.
`resumeExecutionWithResult:`	If a successful, unmatched, invocation of suspendExecution has been made, resume the execution of the command.
`suspendExecution`	Suspends the execution of the receiver.

NSSortDescriptor (New)

Complete reference information is available in the `NSSortDescriptor` reference.

Instance Methods

`ascending`	Returns a Boolean value that indicates whether the receiver specifies sorting in ascending order.
`compareObject:toObject:`	Returns an NSComparisonResult value that indicates the ordering of two given objects.
`initWithKey:ascending:`	Returns an NSSortDescriptor object initialized with a given property key path and sort order, and with the default comparison selector.
`initWithKey:ascending:selector:`	Returns an NSSortDescriptor object initialized with a given property key path, sort order, and comparison selector.
`key`	Returns the receiver’s property key path.
`reversedSortDescriptor`	Returns a copy of the receiver with the sort order reversed.
`selector`	Returns the selector the receiver specifies to use when comparing objects.

NSSpellServer

Complete reference information is available in the `NSSpellServer` reference.

Delegate Methods

`spellServer:suggestCompletionsForPartialWordRange:inString:language:`	This delegate method returns an array of possible word completions from the spell checker, based on a partially completed string and a given range.

NSStream (New)

Complete reference information is available in the `NSStream` reference.

Class Methods

`getStreamsToHost:port:inputStream:outputStream:`	Creates and returns by reference an NSInputStream object and NSOutputStream object for a socket connection with a given host on a given port.

Instance Methods

`close`	Closes the receiver.
`delegate`	Returns the receiver's delegate.
`open`	Opens the receiving stream.
`propertyForKey:`	Returns the receiver's property for a given key.
`removeFromRunLoop:forMode:`	Removes the receiver from a given run loop running in a given mode.
`scheduleInRunLoop:forMode:`	Schedules the receiver on a given run loop in a given mode.
`setDelegate:`	Sets the receiver's delegate.
`setProperty:forKey:`	Attempts to set the value of a given property of the receiver and returns a Boolean value that indicates whether the value is accepted by the receiver.
`streamError`	Returns an NSError object representing the stream error.
`streamStatus`	Returns the receiver's status.

Delegate Methods

`stream:handleEvent:`	The delegate receives this message when a given event has occurred on a given stream.

NSString

Complete reference information is available in the `NSString` reference.

Instance Methods

`getParagraphStart:end:contentsEnd:forRange:`	Returns by reference the beginning of the first paragraph and the end of the last paragraph touched by the given range.
`initWithBytes:length:encoding:`	Returns an initialized NSString object containing a given number of bytes from a given C array of bytes in a given encoding.
`initWithBytesNoCopy:length:encoding:freeWhenDone:`	Returns an initialized NSString object that contains a given number of bytes from a given C array of bytes in a given encoding, and optionally frees the array on deallocation.
`paragraphRangeForRange:`	Returns the range of characters representing the paragraph or paragraphs containing a given range.
`stringByAddingPercentEscapesUsingEncoding:`	Returns a representation of the receiver using a given encoding to determine the percent escapes necessary to convert the receiver into a legal URL string.
`stringByReplacingPercentEscapesUsingEncoding:`	Returns a new string made by replacing in the receiver all percent escapes with the matching characters as determined by a given encoding.

NSURLHandle

Complete reference information is available in the `NSURLHandle` reference.

Instance Methods

`expectedResourceDataSize`	Returns the expected length of the resource data if it is provided by the server.

NSValueTransformer (New)

Complete reference information is available in the `NSValueTransformer` reference.

Class Methods

`allowsReverseTransformation`	Returns a Boolean value that indicates whether the receiver can reverse a transformation.
`setValueTransformer:forName:`	Registers the value transformer a given transformer with a given identifier.
`transformedValueClass`	Returns the class of the value returned by the receiver for a forward transformation.
`valueTransformerForName:`	Returns the value transformer identified by a given identifier.
`valueTransformerNames`	Returns an array of all the registered value transformers.

Instance Methods

`reverseTransformedValue:`	Returns the result of the reverse transformation of a given value.
`transformedValue:`	Returns the result of transforming a given value.

NSXMLParser (New)

Complete reference information is available in the `NSXMLParser` reference.

Instance Methods

`abortParsing`	Stops the parser object.
`columnNumber`	Returns the column number of the XML document being processed by the receiver.
`delegate`	Returns the receiver's delegate.
`initWithContentsOfURL:`	Initializes the receiver with the XML content referenced by the given URL.

`initWithData:`	Initializes the receiver with the XML contents encapsulated in a given data object.
`lineNumber`	Returns the line number of the XML document being processed by the receiver.
`parse`	Starts the event-driven parsing operation.
`parserError`	Returns an NSError object from which you can obtain information about a parsing error.
`publicID`	Returns the public identifier of the external entity referenced in the XML document.
`setDelegate:`	Sets the receiver's delegate.
`setShouldProcessNamespaces:`	Specifies whether the receiver reports the namespace and the qualified name of an element in related delegation methods .
`setShouldReportNamespacePrefixes:`	Specifies whether the receiver reports the scope of namespace declarations using related delegation methods.
`setShouldResolveExternalEntities:`	Specifies whether the receiver reports declarations of external entities using the delegate method parser: foundExternalEntityDeclarationWithName: publicID:systemID:.
`shouldProcessNamespaces`	Indicates whether the receiver reports the namespace and the qualified name of an element in related delegation methods.
`shouldReportNamespacePrefixes`	Indicates whether the receiver reports the prefixes indicating the scope of namespace declarations using related delegation methods.
`shouldResolveExternalEntities`	Indicates whether the receiver reports declarations of external entities using the delegate method parser: foundExternalEntityDeclarationWithName: publicID:systemID:.
`systemID`	Returns the system identifier of the external entity referenced in the XML document.

Delegate Methods

`parser:didEndElement:namespaceURI:qualifiedName:`	Sent by a parser object to its delegate when it encounters an end tag for a specific element.
`parser:didEndMappingPrefix:`	Sent by a parser object to its delegate when a given namespace prefix goes out of scope.

`parser:didStartElement:namespaceURI:qualifiedName:attributes:`	Sent by a parser object to its delegate when it encounters a start tag for a given element.
`parser:didStartMappingPrefix:toURI:`	Sent by a parser object to its delegate the first time it encounters a given namespace prefix, which is mapped to a URI.
`parser:foundAttributeDeclarationWithName:forElement:type:defaultValue:`	Sent by a parser object to its delegate when it encounters a declaration of an attribute that is associated with a specific element.
`parser:foundCDATA:`	Sent by a parser object to its delegate when it encounters a CDATA block.
`parser:foundCharacters:`	Sent by a parser object to provide its delegate with a string representing all or part of the characters of the current element.
`parser:foundComment:`	Sent by a parser object to its delegate when it encounters a comment in the XML.
`parser:foundElementDeclarationWithName:model:`	Sent by a parser object to its delegate when it encounters a declaration of an element with a given model.
`parser:foundExternalEntityDeclarationWithName:publicID:systemID:`	Sent by a parser object to its delegate when it encounters an external entity declaration.
`parser:foundIgnorableWhitespace:`	Reported by a parser object to provide its delegate with a string representing all or part of the ignorable whitespace characters of the current element.
`parser:foundInternalEntityDeclarationWithName:value:`	Sent by a parser object to the delegate when it encounters an internal entity declaration.
`parser:foundNotationDeclarationWithName:publicID:systemID:`	Sent by a parser object to its delegate when it encounters a notation declaration.
`parser:foundProcessingInstructionWithTarget:data:`	Sent by a parser object to its delegate when it encounters a processing instruction.
`parser:foundUnparsedEntityDeclarationWithName:publicID:systemID:notationName:`	Sent by a parser object to its delegate when it encounters an unparsed entity declaration.

`parser:parseErrorOccurred:`	Sent by a parser object to its delegate when it encounters a fatal error.
`parser:resolveExternalEntityName:systemID:`	Sent by a parser object to its delegate when it encounters a given external entity with a specific system ID.
`parser:validationErrorOccurred:`	Sent by a parser object to its delegate when it encounters a fatal validation error. NSXMLParser currently does not invoke this method and does not perform validation.
`parserDidEndDocument:`	Sent by the parser object to the delegate when it has successfully completed parsing.
`parserDidStartDocument:`	Sent by the parser object to the delegate when it begins parsing a document.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

NSAppleEventManager.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSAppleEventManagerSuspensionID`	Identifies an Apple event whose handling has been suspended. Can be used to resume handling of the Apple event.

NSDistributedNotificationCenter.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSNotificationDeliverImmediately`	When set, the notification is delivered immediately to all observers, regardless of their suspension behavior or suspension state. When not set, allows the normal suspension behavior of notification observers to take place.
`NSNotificationPostToAllSessions`	When set, the notification is posted to all sessions. When not set, the notification is sent only to applications within the same login session as the posting task.

NSError.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSUnderlyingErrorKey`	The corresponding value is an error that was encountered in an underlying implementation and caused the error that the receiver represents to occur.

NSKeyValueCoding.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSUndefinedKeyException`	Raised when a key value coding operation fails. userInfo keys are described in “NSUndefinedKeyException userInfo Keys”

NSKeyValueObserving.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSKeyValueChange`	These constants are returned as the value for a NSKeyValueChangeKindKey key in the change dictionary passed to observeValueForKeyPath:ofObject: change:context: indicating the type of change made:

`NSKeyValueChangeIndexesKey`	If the value of the NSKeyValueChangeKindKey entry is NSKeyValueChangeInsertion, NSKeyValueChangeRemoval, or NSKeyValueChangeReplacement, the value of this key is an NSIndexSet object that contains the indexes of the inserted, removed, or replaced objects.
`NSKeyValueChangeInsertion`	Indicates that an object has been inserted into the to-many relationship that is being observed.
`NSKeyValueChangeKindKey`	An NSNumber object that contains a value corresponding to one of the NSKeyValueChangeKindKey enumerations, indicating what sort of change has occurred.
`NSKeyValueChangeNewKey`	If the value of the NSKeyValueChangeKindKey entry is NSKeyValueChangeSetting, and NSKeyValueObservingOptionNew was specified when the observer was registered, the value of this key is the new value for the attribute.
`NSKeyValueChangeOldKey`	If the value of the NSKeyValueChangeKindKey entry is NSKeyValueChangeSetting, and NSKeyValueObservingOptionOld was specified when the observer was registered, the value of this key is the value before the attribute was changed.
`NSKeyValueChangeRemoval`	Indicates that an object has been removed from the to-many relationship that is being observed.
`NSKeyValueChangeReplacement`	Indicates that an object has been replaced in the to-many relationship that is being observed.
`NSKeyValueChangeSetting`	Indicates that the value of the observed key path was set to a new value. This change can occur when observing an attribute of an object, as well as properties that specify to-one and to-many relationships.
`NSKeyValueObservingOptionNew`	Indicates that the change dictionary should provide the new attribute value, if applicable.
`NSKeyValueObservingOptionOld`	Indicates that the change dictionary should contain the old attribute value, if applicable.
`NSKeyValueObservingOptions`	These constants are passed to addObserver:forKeyPath: options:context: and determine the values that are returned as part of the change dictionary passed to an observeValueForKeyPath:ofObject: change:context:. You can pass 0 if you require no change dictionary values.

NSObjCRuntime.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFoundationVersionNumber10_2`	Foundation version released in Mac OS X version 10.2.

NSStream.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSStreamDataWrittenToMemoryStreamKey`	Value is an NSData instance containing the data written to a memory stream.
`NSStreamEvent`	The type declared for the constants listed in "Stream Event Constants."
`NSStreamEventEndEncountered`	The end of the stream has been reached.
`NSStreamEventErrorOccurred`	An error has occurred on the stream.
`NSStreamEventHasBytesAvailable`	The stream has bytes to be read.
`NSStreamEventHasSpaceAvailable`	The stream can accept bytes for writing.
`NSStreamEventNone`	No event has occurred.
`NSStreamEventOpenCompleted`	The open has completed successfully.
`NSStreamFileCurrentOffsetKey`	Value is an NSNumber object containing the current absolute offset of the stream.
`NSStreamSocketSecurityLevelKey`	The security level of the target stream.
`NSStreamSocketSecurityLevelNegotiatedSSL`	Specifies that the highest level security protocol that can be negotiated be set as the security protocol for a socket stream.
`NSStreamSocketSecurityLevelNone`	Specifies that no security level be set for a socket stream.
`NSStreamSocketSecurityLevelSSLv2`	Specifies that SSL version 2 be set as the security protocol for a socket stream.

NSStreamSocketSecurityLevelSSLv3	Specifies that SSL version 3 be set as the security protocol for a socket stream.
NSStreamSocketSecurityLevelTLSv1	Specifies that TLS version 1 be set as the security protocol for a socket stream.
NSStreamSocketSSLErrorDomain	The error domain used by NSError when reporting SSL errors.
NSStreamSOCKSErrorDomain	The error domain used by NSError when reporting SOCKS errors.
NSStreamSOCKSProxyConfigurationKey	Value is an NSDictionary object containing SOCKS proxy configuration information.
NSStreamSOCKSProxyHostKey	Value is an NSString object that represents the SOCKS proxy host.
NSStreamSOCKSProxyPasswordKey	Value is an NSString object containing the user's password.
NSStreamSOCKSProxyPortKey	Value is an NSNumber object containing an integer that represents the port on which the proxy listens.
NSStreamSOCKSProxyUserKey	Value is an NSString object containing the user's name.
NSStreamSOCKSProxyVersion4	Possible value for NSStreamSOCKSProxyVersionKey.
NSStreamSOCKSProxyVersion5	Possible value for NSStreamSOCKSProxyVersionKey.
NSStreamSOCKSProxyVersionKey	Value is either NSStreamSOCKSProxyVersion4 or NSStreamSOCKSProxyVersion5.
NSStreamStatus	The type declared for the constants listed in "Stream Status Constants."
NSStreamStatusAtEnd	There is no more data to read, or no more data can be written to the stream. When this status is returned, the stream is in a "non-blocking" mode and no data are available.
NSStreamStatusClosed	The stream is closed (close has been called on it).
NSStreamStatusError	The remote end of the connection can't be contacted, or the connection has been severed for some other reason.
NSStreamStatusNotOpen	The stream is not open for reading or writing. This status is returned before the underlying call to open a stream but after it's been created.
NSStreamStatusOpen	The stream is open, but no reading or writing is occurring.

`NSStreamStatusOpening`	The stream is in the process of being opened for reading or for writing. For network streams, this status might include the time after the stream was opened, but while network DNS resolution is happening.
`NSStreamStatusReading`	Data is being read from the stream. This status would be returned if code on another thread were to call streamStatus on the stream while a read:maxLength: call (NSInputStream) was in progress.
`NSStreamStatusWriting`	Data is being written to the stream. This status would be returned if code on another thread were to call streamStatus on the stream while a write:maxLength: call (NSOutputStream) was in progress.

NSString.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSNumericSearch`	Numbers within strings are compared using numeric value, that is, Foo2.txt < Foo7.txt < Foo25.txt.

NSURLHandle.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFTPPropertyFTPProxy`	NSDictionary containing proxy information to use in place of proxy identified in SystemConfiguration.framework.

NSValueTransformer.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSIsNilTransformerName	This value transformer returns YES if the value is nil.
NSIsNotNilTransformerName	This value transformer returns YES if the value is non-nil.
NSNegateBooleanTransformerName	This value transformer negates a boolean value, transforming YES to NO and NO to YES.
NSUnarchiveFromDataTransformerName	This value transformer returns an object created by attempting to unarchive the data in the NSData object passed as the value.

NSXMLParser.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSXMLParserAttributeHasNoValueError	Attribute doesn't contain a value.
NSXMLParserAttributeListNotFinishedError	Attribute list is not finished.
NSXMLParserAttributeListNotStartedError	Attribute list is not started.
NSXMLParserAttributeNotFinishedError	Attribute is not finished.
NSXMLParserAttributeNotStartedError	Attribute is not started.
NSXMLParserAttributeRedefinedError	Attribute is redefined.
NSXMLParserCDATANotFinishedError	CDATA block is not finished.
NSXMLParserCharacterRefAtEOFError	Target of character reference cannot be found.
NSXMLParserCharacterRefInDTDError	Invalid character encountered in the DTD.
NSXMLParserCharacterRefInEpilogError	Invalid character found in the epilog.
NSXMLParserCharacterRefInPrologError	Invalid character found in the prolog.
NSXMLParserCommentContainsDoubleHyphenError	Comment contains double hyphen.
NSXMLParserCommentNotFinishedError	Comment is not finished.
NSXMLParserConditionalSectionNotFinishedError	Conditional section is not finished.
NSXMLParserConditionalSectionNotStartedError	Conditional section is not started.
NSXMLParserDelegateAbortedParseError	Delegate aborted parse.

NSXMLParserDOCTYPEDeclNotFinishedError	Document type declaration is not finished.
NSXMLParserDocumentStartError	The parser object is unable to start parsing.
NSXMLParserElementContentDeclNotFinishedError	Element content declaration is not finished.
NSXMLParserElementContentDeclNotStartedError	Element content declaration is not started.
NSXMLParserEmptyDocumentError	The document is empty.
NSXMLParserEncodingNotSupportedError	Document encoding is not supported.
NSXMLParserEntityBoundaryError	Entity boundary error.
NSXMLParserEntityIsExternalError	Cannot parse external entity.
NSXMLParserEntityIsParameterError	Entity is a parameter.
NSXMLParserEntityNotFinishedError	Entity is not finished.
NSXMLParserEntityNotStartedError	Entity is not started.
NSXMLParserEntityRefAtEOFError	Target of entity reference is not found.
NSXMLParserEntityReferenceMissingSemiError	Entity reference is missing semicolon.
NSXMLParserEntityReferenceWithoutNameError	Entity reference is without name.
NSXMLParserEntityRefInDTDError	Invalid entity reference found in the DTD.
NSXMLParserEntityRefInEpilogError	Invalid entity reference found in the epilog.
NSXMLParserEntityRefInPrologError	Invalid entity reference found in the prolog.
NSXMLParserEntityRefLoopError	Entity reference loop encountered.
NSXMLParserEntityValueRequiredError	Entity value is required.
NSXMLParserEqualExpectedError	Equal sign expected.
NSXMLParserError	A type defined for the contants listed in “Parser Error Constants.”
NSXMLParserErrorDomain	Indicates an error in XML parsing.
NSXMLParserExternalStandaloneEntityError	External standalone entity.
NSXMLParserExternalSubsetNotFinishedError	External subset is not finished.
NSXMLParserExtraContentError	Error in content found.

NSXMLParserGTRequiredError	Right angle bracket is required.
NSXMLParserInternalError	The parser object encountered an internal error.
NSXMLParserInvalidCharacterError	Invalid character encountered.
NSXMLParserInvalidCharacterInEntityError	Invalid character in entity found.
NSXMLParserInvalidCharacterRefError	Invalid character reference encountered.
NSXMLParserInvalidConditionalSectionError	Invalid conditional section.
NSXMLParserInvalidDecimalCharacterRefError	Invalid decimal character reference encountered.
NSXMLParserInvalidEncodingError	Invalid encoding.
NSXMLParserInvalidEncodingNameError	Invalid encoding name found.
NSXMLParserInvalidHexCharacterRefError	Invalid hexadecimal character reference encountered.
NSXMLParserInvalidURIError	Invalid URI specified.
NSXMLParserLessThanSymbolInAttributeError	Angle bracket is used in attribute.
NSXMLParserLiteralNotFinishedError	Literal is not finished.
NSXMLParserLiteralNotStartedError	Literal is not started.
NSXMLParserLTRequiredError	Left angle bracket is required.
NSXMLParserLTSlashRequiredError	Left angle bracket slash is required.
NSXMLParserMisplacedCDATAEndStringError	Misplaced CDATA end string.
NSXMLParserMisplacedXMLDeclarationError	Misplaced XML declaration.
NSXMLParserMixedContentDeclNotFinishedError	Mixed content declaration is not finished.
NSXMLParserMixedContentDeclNotStartedError	Mixed content declaration is not started.
NSXMLParserNAMERequiredError	Name is required.
NSXMLParserNamespaceDeclarationError	Invalid namespace declaration encountered.
NSXMLParserNMTOKENRequiredError	Name token is required.
NSXMLParserNoDTDError	Missing DTD.
NSXMLParserNotationNotFinishedError	Notation is not finished.
NSXMLParserNotationNotStartedError	Notation is not started.

NSXMLParserNotWellBalancedError	Document is not well balanced.
NSXMLParserOutOfMemoryError	The parser object ran out of memory.
NSXMLParserParsedEntityRefAtEOFError	Target of parsed entity reference is not found.
NSXMLParserParsedEntityRefInEpilogError	Target of parsed entity reference is not found in epilog.
NSXMLParserParsedEntityRefInInternalError	Internal error in parsed entity reference found.
NSXMLParserParsedEntityRefInInternalSubsetError	Target of parsed entity reference is not found in internal subset.
NSXMLParserParsedEntityRefInPrologError	Target of parsed entity reference is not found in prolog.
NSXMLParserParsedEntityRefMissingSemiError	Parsed entity reference is missing semicolon.
NSXMLParserParsedEntityRefNoNameError	Parsed entity reference is without an entity name.
NSXMLParserPCDATARequiredError	CDATA is required.
NSXMLParserPrematureDocumentEndError	The document ended unexpectedly.
NSXMLParserProcessingInstructionNotFinishedError	Processing instruction is not finished.
NSXMLParserProcessingInstructionNotStartedError	Processing instruction is not started.
NSXMLParserPublicIdentifierRequiredError	Public identifier is required.
NSXMLParserSeparatorRequiredError	Separator is required.
NSXMLParserSpaceRequiredError	Space is required.
NSXMLParserStandaloneValueError	Standalone value found.
NSXMLParserStringNotClosedError	String is not closed.
NSXMLParserStringNotStartedError	String is not started.
NSXMLParserTagNameMismatchError	Tag name mismatch.
NSXMLParserUndeclaredEntityError	Entity is not declared.
NSXMLParserUnfinishedTagError	Unfinished tag found.
NSXMLParserUnknownEncodingError	Document encoding is unknown.
NSXMLParserUnparsedEntityError	Cannot parse entity.
NSXMLParserURIFragmentError	URI fragment.

`NSXMLParserURIRequiredError`	URI is required.
`NSXMLParserXMLDeclNotFinishedError`	XML declaration is not finished.
`NSXMLParserXMLDeclNotStartedError`	XML declaration is not started.

10.2 Symbol Changes

This article lists the symbols added to `Foundation.framework` in Mac OS X v10.2.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

NSAppleEventDescriptor

Complete reference information is available in the `NSAppleEventDescriptor` reference.

Class Methods

`descriptorWithBoolean:`	Creates a descriptor initialized with type typeBoolean that stores the specified Boolean value.
`descriptorWithDescriptorType:bytes:length:`	Creates a descriptor initialized with the specified event type that stores the specified data (from a series of bytes).
`descriptorWithEnumCode:`	Creates a descriptor initialized with type typeEnumerated that stores the specified enumerator data type value.
`descriptorWithInt32:`	Creates a descriptor initialized with Apple event type typeSInt32 that stores the specified integer value.
`descriptorWithString:`	Creates a descriptor initialized with type typeUnicodeText that stores the text from the specified string.
`descriptorWithTypeCode:`	Creates a descriptor initialized with type typeType that stores the specified type value.

Instance Methods

`aeDesc`	Returns a pointer to the AEDesc structure that is encapsulated by the receiver, if it has one.
`booleanValue`	Returns the contents of the receiver as a Boolean value, coercing (to typeBoolean) if necessary.
`enumCodeValue`	Returns the contents of the receiver as an enumeration type, coercing (to typeEnumerated) if necessary.
`initWithAEDescNoCopy:`	Initializes a newly allocated instance as a descriptor for the specified Carbon AEDesc structure.
`initWithDescriptorType:bytes:length:`	Initializes a newly allocated instance as a descriptor with the specified descriptor type and data (from an arbitrary sequence of bytes and a length count).
`int32Value`	Returns the contents of the receiver as an integer, coercing (to typeSInt32) if necessary.
`removeDescriptorAtIndex:`	Removes the descriptor at the specified (one-based) position in the receiving descriptor list.
`stringValue`	Returns the contents of the receiver as a Unicode text string, coercing (to typeUnicodeText) if necessary.
`typeCodeValue`	Returns the contents of the receiver as a type, coercing (to typeType) if necessary.

NSAppleScript (New)

Complete reference information is available in the `NSAppleScript` reference.

Instance Methods

`compileAndReturnError:`	Compiles the receiver, if it is not already compiled.
`executeAndReturnError:`	Executes the receiver, compiling it first if it is not already compiled.
`executeAppleEvent:error:`	Executes an Apple event in the context of the receiver, as a means of allowing the application to invoke a handler in the script.
`initWithContentsOfURL:error:`	Initializes a newly allocated script instance from the source identified by the passed URL.
`initWithSource:`	Initializes a newly allocated script instance from the passed source.
`isCompiled`	Returns a Boolean value that indicates whether the receiver's script has been compiled.

`source`	Returns the script source for the receiver.

NSArray

Complete reference information is available in the `NSArray` reference.

Instance Methods

`initWithArray:copyItems:`	Initializes a newly allocated array using anArray as the source of data objects for the array.

NSBundle

Complete reference information is available in the `NSBundle` reference.

Class Methods

`preferredLocalizationsFromArray:forPreferences:`	Returns the localizations that a bundle object would prefer, given the specified bundle and user preference localizations.

Instance Methods

`developmentLocalization`	Returns the localization used to create the bundle.
`isLoaded`	Obtains information about the load status of a bundle.
`localizedInfoDictionary`	Returns a dictionary with the keys from the bundle's localized property list.
`objectForInfoDictionaryKey:`	Returns the value associated with a given key in the receiver's property list.

NSCachedURLResponse (New)

Complete reference information is available in the `NSCachedURLResponse` reference.

Instance Methods

`data`	Returns the receiver's cached data.
`initWithResponse:data:`	Initializes an NSCachedURLResponse object.

`initWithResponse:data:userInfo:storagePolicy:`	Initializes an NSCachedURLResponse object.
`response`	Returns the NSURLResponse object associated with the receiver.
`storagePolicy`	Returns the receiver's cache storage policy.
`userInfo`	Returns the receiver's user info dictionary.

NSCharacterSet

Complete reference information is available in the `NSCharacterSet` reference.

Class Methods

`capitalizedLetterCharacterSet`	Returns a character set containing the characters in the category of Titlecase Letters.

Instance Methods

`hasMemberInPlane:`	Returns a Boolean value that indicates whether the receiver has at least one member in a given character plane.
`isSupersetOfSet:`	Returns a Boolean value that indicates whether the receiver is a superset of another given character set.
`longCharacterIsMember:`	Returns a Boolean value that indicates whether a given long character is a member of the receiver.

NSCoder

Complete reference information is available in the `NSCoder` reference.

Instance Methods

`allowsKeyedCoding`	Returns a Boolean value that indicates whether the receiver supports keyed coding of objects.
`containsValueForKey:`	Returns a Boolean value that indicates whether an encoded value is available for a string.
`decodeBoolForKey:`	Decodes and returns a boolean value that was previously encoded with encodeBool:forKey: and associated with the string key.

`decodeBytesForKey:returnedLength:`	Decodes a buffer of data that was previously encoded with encodeBytes:length:forKey: and associated with the string key.
`decodeDoubleForKey:`	Decodes and returns a double value that was previously encoded with either encodeFloat:forKey: or encodeDouble:forKey: and associated with the string key.
`decodeFloatForKey:`	Decodes and returns a float value that was previously encoded with encodeFloat:forKey: or encodeDouble:forKey: and associated with the string key.
`decodeInt32ForKey:`	Decodes and returns a 32-bit integer value that was previously encoded with encodeInt:forKey:, encodeInt32:forKey:, or encodeInt64:forKey: and associated with the string key.
`decodeInt64ForKey:`	Decodes and returns a 64-bit integer value that was previously encoded with encodeInt:forKey:, encodeInt32:forKey:, or encodeInt64:forKey: and associated with the string key.
`decodeIntForKey:`	Decodes and returns an int value that was previously encoded with encodeInt:forKey:, encodeInt32:forKey:, or encodeInt64:forKey: and associated with the string key.
`decodeObjectForKey:`	Decodes and returns an autoreleased Objective-C object that was previously encoded with encodeObject:forKey: or encodeConditionalObject:forKey: and associated with the string key.
`decodePointForKey:`	Decodes and returns an NSPoint structure that was previously encoded with encodePoint:forKey:.
`decodeRectForKey:`	Decodes and returns an NSRect structure that was previously encoded with encodeRect:forKey:.
`decodeSizeForKey:`	Decodes and returns an NSSize structure that was previously encoded with encodeSize:forKey:.
`encodeBool:forKey:`	Encodes boolv and associates it with the string key.
`encodeBytes:length:forKey:`	Encodes a buffer of data, bytesp, whose length is specified by lenv, and associates it with the string key.
`encodeConditionalObject:forKey:`	Conditionally encodes a reference to objv and associates it with the string key only if objv has been unconditionally encoded with encodeObject:forKey:.
`encodeDouble:forKey:`	Encodes realv and associates it with the string key.
`encodeFloat:forKey:`	Encodes realv and associates it with the string key.
`encodeInt32:forKey:`	Encodes the 32-bit integer intv and associates it with the string key.

`encodeInt64:forKey:`	Encodes the 64-bit integer intv and associates it with the string key.
`encodeInt:forKey:`	Encodes intv and associates it with the string key.
`encodeObject:forKey:`	Encodes the object objv and associates it with the string key.
`encodePoint:forKey:`	Encodes point and associates it with the string key.
`encodeRect:forKey:`	Encodes rect and associates it with the string key.
`encodeSize:forKey:`	Encodes size and associates it with the string key.

NSData

Complete reference information is available in the `NSData` reference.

Class Methods

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

Instance Methods

`initWithBytesNoCopy:length:freeWhenDone:`	Initializes a newly allocated data object by adding to it length bytes of data from the buffer bytes.

NSDictionary

Complete reference information is available in the `NSDictionary` reference.

Instance Methods

`fileCreationDate`	Returns the value for the NSFileCreationDate key.
`fileGroupOwnerAccountID`	Returns the value for the NSFileGroupOwnerAccountID key.
`fileIsAppendOnly`	Returns the value for the NSFileAppendOnly key.
`fileIsImmutable`	Returns the value for the NSFileImmutable key.
`fileOwnerAccountID`	Returns the value for the NSFileOwnerAccountID key.

NSError (New)

Complete reference information is available in the `NSError` reference.

Class Methods

`errorWithDomain:code:userInfo:`	Creates and initializes an NSError object for a given domain and code with a given userInfo dictionary.

Instance Methods

`code`	Returns the receiver's error code.
`domain`	Returns the receiver's error domain.
`initWithDomain:code:userInfo:`	Returns an NSError object initialized for a given domain and code with a given userInfo dictionary.
`localizedDescription`	Returns a string containing the localized description of the error.
`userInfo`	Returns the receiver's user info dictionary.

NSFileManager

Complete reference information is available in the `NSFileManager` reference.

Instance Methods

`componentsToDisplayForPath:`	Returns an array of NSString objects representing the user-visible components of a given path.

NSHTTPCookie (New)

Complete reference information is available in the `NSHTTPCookie` reference.

Class Methods

`cookiesWithResponseHeaderFields:forURL:`	Returns an array of NSHTTPCookie objects corresponding to the provided response header fields for the provided URL.
`cookieWithProperties:`	Creates and initializes an NSHTTPCookie object using the provided properties.

`requestHeaderFieldsWithCookies:`	Returns a dictionary of header fields corresponding to a provided array of cookies.

Instance Methods

`comment`	Returns the receiver's comment string.
`commentURL`	Returns the receiver’s comment URL.
`domain`	Returns the domain of the receiver’s cookie.
`expiresDate`	Returns the receiver’s expiration date.
`initWithProperties:`	Returns an initialized NSHTTPCookie object using the provided properties.
`isSecure`	Returns whether his cookie should only be sent over secure channels.
`isSessionOnly`	Returns whether the receiver should be discarded at the end of the session (regardless of expiration date).
`name`	Returns the receiver’s name.
`path`	Returns the receiver’s path.
`portList`	Returns the receiver's port list.
`properties`	Returns the receiver’s cookie properties.
`value`	Returns the receiver’s value.
`version`	Returns the receiver’s version.

NSHTTPCookieStorage (New)

Complete reference information is available in the `NSHTTPCookieStorage` reference.

Class Methods

`sharedHTTPCookieStorage`	Returns the shared cookie storage instance.

Instance Methods

`cookieAcceptPolicy`	Returns the receiver’s cookie accept policy.
`cookies`	Returns the receiver’s cookies.

`cookiesForURL:`	Returns all the receiver's cookies that will be sent to a specified URL.
`deleteCookie:`	Deletes the specified cookie from the receiver.
`setCookie:`	Stores a specified cookie in the receiver if the receiver's cookie accept policy permits.
`setCookieAcceptPolicy:`	Sets the cookie accept policy of the receiver
`setCookies:forURL:mainDocumentURL:`	Adds an array of cookies to the receiver if the receiver's cookie acceptance policy permits.

NSHTTPURLResponse (New)

Complete reference information is available in the `NSHTTPURLResponse` reference.

Class Methods

`localizedStringForStatusCode:`	Returns a localized string corresponding to a specified HTTP status code.

Instance Methods

`allHeaderFields`	Returns all the HTTP header fields of the receiver.
`statusCode`	Returns the receiver's HTTP status code.

NSKeyedArchiver (New)

Complete reference information is available in the `NSKeyedArchiver` reference.

Class Methods

`archivedDataWithRootObject:`	Returns an NSData object containing the encoded form of the object graph whose root object is given.
`archiveRootObject:toFile:`	Archives an object graph rooted at a given object by encoding it into a data object then atomically writes the resulting data object to a file at a given path, and returns a Boolean value that indicates whether the operation was successful.
`classNameForClass:`	Returns the class name with which NSKeyedArchiver encodes instances of a given class.

`setClassName:forClass:`	Adds a class translation mapping to NSKeyedArchiver whereby instances of of a given class are encoded with a given class name instead of their real class names.

Instance Methods

`classNameForClass:`	Returns the class name with which the receiver encodes instances of a given class.
`delegate`	Returns the receiver's delegate.
`encodeBool:forKey:`	Encodes a given Boolean value and associates it with a given key.
`encodeBytes:length:forKey:`	Encodes a given number of bytes from a given C array of bytes and associates them with the a given key.
`encodeConditionalObject:forKey:`	Encodes a reference to a given object and associates it with a given key only if it has been unconditionally encoded elsewhere in the archive with encodeObject:forKey:.
`encodeDouble:forKey:`	Encodes a given double value and associates it with a given key.
`encodeFloat:forKey:`	Encodes a given float value and associates it with a given key.
`encodeInt32:forKey:`	Encodes a given 32-bit integer value and associates it with a given key.
`encodeInt64:forKey:`	Encodes a given 64-bit integer value and associates it with a given key.
`encodeInt:forKey:`	Encodes a given int value and associates it with a given key.
`encodeObject:forKey:`	Encodes a given object and associates it with a given key.
`finishEncoding`	Instructs the receiver to construct the final data stream.
`initForWritingWithMutableData:`	Returns the receiver, initialized for encoding an archive into a given a mutable-data object.
`outputFormat`	Returns the format in which the receiver encodes its data.
`setClassName:forClass:`	Adds a class translation mapping to the receiver whereby instances of of a given class are encoded with a given class name instead of their real class names.
`setDelegate:`	Sets the delegate for the receiver.
`setOutputFormat:`	Sets the format in which the receiver encodes its data.

Delegate Methods

`archiver:didEncodeObject:`	Informs the delegate that a given object has been encoded.
`archiver:willEncodeObject:`	Informs the delegate that object is about to be encoded.
`archiver:willReplaceObject:withObject:`	Informs the delegate that one given object is being substituted for another given object.
`archiverDidFinish:`	Notifies the delegate that encoding has finished.
`archiverWillFinish:`	Notifies the delegate that encoding is about to finish.
`replacementObjectForKeyedArchiver:`	Overridden by subclasses to substitute another object for itself during keyed archiving.

NSKeyedUnarchiver (New)

Complete reference information is available in the `NSKeyedUnarchiver` reference.

Class Methods

`classForClassName:`	Returns the class from which NSKeyedUnarchiver instantiates an encoded object with a given class name.
`setClass:forClassName:`	Adds a class translation mapping to NSKeyedUnarchiver whereby objects encoded with a given class name are decoded as instances of a given class instead.
`unarchiveObjectWithData:`	Decodes and returns the object graph previously encoded by NSKeyedArchiver and stored in a given NSData object.
`unarchiveObjectWithFile:`	Decodes and returns the object graph previously encoded by NSKeyedArchiver written to the file at a given path.

Instance Methods

`classForClassName:`	Returns the class from which the receiver instantiates an encoded object with a given class name.
`containsValueForKey:`	Returns a Boolean value that indicates whether the archive contains a value for a given key within the current decoding scope.
`decodeBoolForKey:`	Decodes a Boolean value associated with a given key.
`decodeBytesForKey:returnedLength:`	Decodes a stream of bytes associated with a given key.

`decodeDoubleForKey:`	Decodes a double-precision floating-point value associated with a given key.
`decodeFloatForKey:`	Decodes a single-precision floating-point value associated with a given key.
`decodeInt32ForKey:`	Decodes a 32-bit integer value associated with a given key.
`decodeInt64ForKey:`	Decodes a 64-bit integer value associated with a given key.
`decodeIntForKey:`	Decodes an integer value associated with a given key.
`decodeObjectForKey:`	Decodes and returns an object associated with a given key.
`delegate`	Returns the receiver's delegate.
`finishDecoding`	Tells the receiver that you are finished decoding objects.
`initForReadingWithData:`	Initializes the receiver for decoding an archive previously encoded by NSKeyedArchiver.
`setClass:forClassName:`	Adds a class translation mapping to the receiver whereby objects encoded with a given class name are decoded as instances of a given class instead.
`setDelegate:`	Sets the receiver's delegate.

Delegate Methods

`unarchiver:cannotDecodeObjectOfClassName:originalClasses:`	Informs the delegate that the class with a given name is not available during decoding.
`unarchiver:didDecodeObject:`	Informs the delegate that a given object has been decoded.
`unarchiver:willReplaceObject:withObject:`	Informs the delegate that one object is being substituted for another.
`unarchiverDidFinish:`	Notifies the delegate that decoding has finished.
`unarchiverWillFinish:`	Notifies the delegate that decoding is about to finish.

NSMutableArray

Complete reference information is available in the `NSMutableArray` reference.

Instance Methods

`exchangeObjectAtIndex:withObjectAtIndex:`	Exchanges the objects in the receiver at given indices.

NSMutableData

Complete reference information is available in the `NSMutableData` reference.

Instance Methods

`replaceBytesInRange:withBytes:length:`	Replaces with a given set of bytes a given range within the contents of the receiver.

NSMutableString

Complete reference information is available in the `NSMutableString` reference.

Instance Methods

`replaceOccurrencesOfString:withString:options:range:`	Replaces all occurrences of a given string in a given range with another given string, returning the number of replacements.

NSMutableURLRequest (New)

Complete reference information is available in the `NSMutableURLRequest` reference.

Instance Methods

`addValue:forHTTPHeaderField:`	Adds an HTTP header to the receiver's HTTP header dictionary.
`setAllHTTPHeaderFields:`	Replaces the receiver's header fields with the passed values.
`setCachePolicy:`	Sets the cache policy of the receiver.
`setHTTPBody:`	Sets the request body of the receiver to the specified data.
`setHTTPMethod:`	Sets the receiver's HTTP request method.
`setHTTPShouldHandleCookies:`	Sets whether the receiver should use the default cookie handling for the request.
`setMainDocumentURL:`	Sets the main document URL for the receiver.

`setTimeoutInterval:`	Sets the receiver's timeout interval, in seconds.
`setURL:`	Sets the URL of the receiver
`setValue:forHTTPHeaderField:`	Sets the specified HTTP header field.

NSNameSpecifier (New)

Complete reference information is available in the `NSNameSpecifier` reference.

Instance Methods

`initWithContainerClassDescription: containerSpecifier:key:name:`	Invokes the super class's initWithContainerClassDescription: containerSpecifier:key: method and then sets the name instance variable to name.
`name`	Returns the name encapsulated by the receiver for the specified object in the container.
`setName:`	Sets the name encapsulated with the receiver for the specified object in the container.

NSNetService (New)

Complete reference information is available in the `NSNetService` reference.

Instance Methods

`addresses`	Returns an array containing NSData objects, each of which contains a socket address for the service.
`delegate`	Returns the delegate for the receiver.
`domain`	Returns the domain name of the service.
`initWithDomain:type:name:`	Returns the receiver, initialized as a network service of a given type and sets the initial host information.
`initWithDomain:type:name:port:`	Initializes the receiver as a network service of type type at the socket location specified by domain, name, and port.
`name`	Returns the name of the service.
`protocolSpecificInformation`	Returns protocol specific information for legacy ZeroConf-style clients.
`publish`	Attempts to advertise the receiver's on the network.

`removeFromRunLoop:forMode:`	Removes the service from the given run loop for a given mode.
`resolve`	Starts a resolve process for the receiver.
`scheduleInRunLoop:forMode:`	Adds the service to the specified run loop.
`setDelegate:`	Sets the delegate for the receiver.
`setProtocolSpecificInformation:`	Sets protocol specific information for legacy ZeroConf-style clients.
`stop`	Halts a currently running attempt to publish or resolve a service.
`type`	Returns the type of the service.

Delegate Methods

`netService:didNotPublish:`	Notifies the delegate that a service could not be published.
`netService:didNotResolve:`	Informs the delegate that an error occurred during resolution of a given service.
`netServiceDidResolveAddress:`	Informs the delegate that the address for a given service was resolved.
`netServiceDidStop:`	Informs the delegate that a publish or resolveWithTimeout: request was stopped.
`netServiceWillPublish:`	Notifies the delegate that the network is ready to publish the service.
`netServiceWillResolve:`	Notifies the delegate that the network is ready to resolve the service.

NSNetServiceBrowser (New)

Complete reference information is available in the `NSNetServiceBrowser` reference.

Instance Methods

`delegate`	Returns the receiver's delegate.
`init`	Initializes an allocated NSNetServiceBrowser object.
`removeFromRunLoop:forMode:`	Removes the receiver from the specified run loop.
`scheduleInRunLoop:forMode:`	Adds the receiver to the specified run loop.
`searchForAllDomains`	Initiates a search for all domains that are visible to the host.

`searchForRegistrationDomains`	Initiates a search for domains in which the host may register services.
`searchForServicesOfType:inDomain:`	Starts a search for services of a particular type within a specific domain.
`setDelegate:`	Sets the receiver's delegate.
`stop`	Halts a currently running search or resolution.

Delegate Methods

`netServiceBrowser:didFindDomain:moreComing:`	Tells the delegate the sender found a domain.
`netServiceBrowser:didFindService:moreComing:`	Tells the delegate the sender found a service.
`netServiceBrowser:didNotSearch:`	Tells the delegate that a search was not successful.
`netServiceBrowser:didRemoveDomain:moreComing:`	Tells the delegate the a domain has disappeared or has become unavailable.
`netServiceBrowser:didRemoveService:moreComing:`	Tells the delegate a service has disappeared or has become unavailable.
`netServiceBrowserDidStopSearch:`	Tells the delegate that a search was stopped.
`netServiceBrowserWillSearch:`	Tells the delegate that a serch is commencing.

NSObject

Complete reference information is available in the `NSObject` reference.

Class Methods

`cancelPreviousPerformRequestsWithTarget:`	Cancels perform requests previously registered with the performSelector:withObject: afterDelay: instance method.
`classForKeyedUnarchiver`	Overridden by subclasses to substitute a new class during keyed unarchiving.
`isSubclassOfClass:`	Returns a Boolean value that indicates whether the receiving class is a subclass of, or identical to, a given class.

Instance Methods

`classForKeyedArchiver`	Overridden by subclasses to substitute a new class for instances during keyed archiving.
`insertValue:inPropertyWithKey:`	Inserts an object in the collection specified by the passed key.
`performSelectorOnMainThread:withObject:waitUntilDone:`	Invokes a method of the receiver on the main thread using the default mode.
`performSelectorOnMainThread:withObject:waitUntilDone:modes:`	Invokes a method of the receiver on the main thread using the specified modes.
`scriptingProperties`	Returns an NSString-keyed dictionary of the receiver's scriptable properties.
`setScriptingProperties:`	Given an NSString-keyed dictionary, sets one or more scriptable properties of the receiver.
`valueWithName:inPropertyWithKey:`	Retrieves a named object from the collection specified by the passed key.
`valueWithUniqueID:inPropertyWithKey:`	Retrieves an object by ID from the collection specified by the passed key.

NSPositionalSpecifier

Complete reference information is available in the `NSPositionalSpecifier` reference.

Instance Methods

`insertionReplaces`	Returns a Boolean value that indicates whether evaluation has been successful and the object to be inserted should actually replace the keyed, indexed object in the insertion container.
`setInsertionClassDescription:`	Sets the class description for the object or objects to be inserted.

NSProcessInfo

Complete reference information is available in the `NSProcessInfo` reference.

Instance Methods

`operatingSystemVersionString`	Returns a string containing the version of the operating system on which the process is executing.

NSPropertyListSerialization (New)

Complete reference information is available in the `NSPropertyListSerialization` reference.

Class Methods

`dataFromPropertyList:format:errorDescription:`	Returns an NSData object containing a given property list in a specified format.
`propertyList:isValidForFormat:`	Returns a Boolean value that indicates whether a given property list is valid for a given format.
`propertyListFromData:mutabilityOption:format:errorDescription:`	Returns a property list object corresponding to the representation in a given NSData object.

NSRunLoop

Complete reference information is available in the `NSRunLoop` reference.

Instance Methods

`cancelPerformSelectorsWithTarget:`	Cancels all outstanding ordered performs scheduled with a given target.

NSScriptClassDescription

Complete reference information is available in the `NSScriptClassDescription` reference.

Instance Methods

`defaultSubcontainerAttributeKey`	Returns the value of the DefaultSubcontainerAttribute entry of the class dictionary from which the receiver was instantiated.
`isLocationRequiredToCreateForKey:`	Returns a Boolean value indicating whether an insertion location must be specified when creating a new object in the specified to-many relationship of the receiver.

NSSocketPortNameServer

Complete reference information is available in the `NSSocketPortNameServer` reference.

Instance Methods

`registerPort:name:`	Registers a given port as a network service with the specified name in the local domain.

NSString

Complete reference information is available in the `NSString` reference.

Instance Methods

`decomposedStringWithCanonicalMapping`	Returns a string made by normalizing the receiver's contents using Form D.
`decomposedStringWithCompatibilityMapping`	Returns a string made by normalizing the receiver's contents using Form KD.
`precomposedStringWithCanonicalMapping`	Returns a string made by normalizing the receiver's contents using Form C.
`precomposedStringWithCompatibilityMapping`	Returns a string made by normalizing the receiver's contents using Form KC.
`stringByPaddingToLength:withString: startingAtIndex:`	Returns a new string formed from the receiver by either removing characters from the end, or by appending as many occurrences as necessary of a given pad string.
`stringByTrimmingCharactersInSet:`	Returns a new string made by removing from both ends of the receiver characters contained in a given character set.

NSThread

Complete reference information is available in the `NSThread` reference.

Class Methods

`setThreadPriority:`	Sets the current thread's priority.
`threadPriority`	Returns the current thread's priority.

NSTimer

Complete reference information is available in the `NSTimer` reference.

Instance Methods

`initWithFireDate:interval:target:selector:userInfo:repeats:`	Initializes a new NSTimer that, when added to a run loop, will fire at date and then, if repeats is YES, every seconds after that.
`setFireDate:`	Resets the receiver to fire next at a given date.

NSUniqueIDSpecifier (New)

Complete reference information is available in the `NSUniqueIDSpecifier` reference.

Instance Methods

`initWithContainerClassDescription:containerSpecifier:key:uniqueID:`	Returns an NSUniqueIDSpecifier object, initialized with the given arguments.
`setUniqueID:`	Sets the ID encapsulated by the receiver.
`uniqueID`	Returns the ID encapsulated by the receiver.

NSURLAuthenticationChallenge (New)

Complete reference information is available in the `NSURLAuthenticationChallenge` reference.

Instance Methods

`error`	Returns the NSError object representing the last authentication failure.
`failureResponse`	Returns the NSURLResponse object representing the last authentication failure.
`initWithAuthenticationChallenge:sender:`	Returns an initialized NSURLAuthenticationChallenge object copying the properties from challenge, and setting the authentication sender to sender.
`initWithProtectionSpace:proposedCredential:previousFailureCount:failureResponse:error:sender:`	Returns an initialized NSURLAuthenticationChallenge object for the specified space using the credential, or nil if there is no proposed credential.

previousFailureCount	Returns the receiver's count of failed authentication attempts.
proposedCredential	Returns the proposed credential for this challenge.
protectionSpace	Returns the receiver's protection space.
sender	Returns the receiver's sender.

NSURLCache (New)

Complete reference information is available in the NSURLCache reference.

Class Methods

setSharedURLCache:	Sets the shared NSURLCache instance to a specified cache object.
sharedURLCache	Returns the shared NSURLCache instance.

Instance Methods

cachedResponseForRequest:	Returns the cached URL response in the cache for the specified URL request.
currentDiskUsage	Returns the current size of the receiver's on-disk cache, in bytes.
currentMemoryUsage	Returns the current size of the receiver's in-memory cache, in bytes.
diskCapacity	Returns the capacity of the receiver's on-disk cache, in bytes.
initWithMemoryCapacity:diskCapacity:diskPath:	Initializes an NSURLCache object with the specified values.
memoryCapacity	Returns the capacity of the receiver's in-memory cache, in bytes.
removeAllCachedResponses	Clears the receiver's cache, removing all stored cached URL responses.
removeCachedResponseForRequest:	Removes the cached URL response for a specified URL request.
setDiskCapacity:	Sets the receiver's on-disk cache capacity
setMemoryCapacity:	Sets the receiver's in-memory cache capacity.

`storeCachedResponse:forRequest:`	Stores a cached URL response for a specified request

NSURLConnection (New)

Complete reference information is available in the `NSURLConnection` reference.

Class Methods

`canHandleRequest:`	Returns whether a request can be handled based on a "preflight" evaluation.
`connectionWithRequest:delegate:`	Creates and returns an initialized URL connection and begins to load the data for the URL request.
`sendSynchronousRequest:returningResponse:error:`	Performs a synchronous load of the specified URL request.

Instance Methods

`cancel`	Cancels an asynchronous load of a request.
`initWithRequest:delegate:`	Returns an initialized URL connection and begins to load the data for the URL request.

Delegate Methods

`connection:didCancelAuthenticationChallenge:`	Sent when a connection cancels an authentication challenge.
`connection:didFailWithError:`	Sent when a connection fails to load its request successfully.
`connection:didReceiveAuthenticationChallenge:`	Sent when a connection must authenticate a challenge in order to download its request.
`connection:didReceiveData:`	Sent as a connection loads data incrementally.
`connection:didReceiveResponse:`	Sent when the connection has received sufficient data to construct the URL response for its request.
`connection:willCacheResponse:`	Sent before the connection stores a cached response in the cache, to give the delegate an opportunity to alter it.

`connection:willSendRequest:redirectResponse:`	Sent when the connection determines that it must change URLs in order to continue loading a request.
`connectionDidFinishLoading:`	Sent when a connection has finished loading successfully.

NSURLCredential (New)

Complete reference information is available in the `NSURLCredential` reference.

Class Methods

`credentialWithUser:password:persistence:`	Creates and returns an NSURLCredential object with a given user name and password using a given persistence setting.

Instance Methods

`hasPassword`	Returns a Boolean value that indicates whether the receiver has a password.
`initWithUser:password:persistence:`	Returns an NSURLCredential object initialized with a given user name and password using a given persistence setting.
`password`	Returns the receiver's password.
`persistence`	Returns the receiver's persistence setting.
`user`	Returns the receiver's user name.

NSURLCredentialStorage (New)

Complete reference information is available in the `NSURLCredentialStorage` reference.

Class Methods

`sharedCredentialStorage`	Returns the shared URL credential storage object.

Instance Methods

`allCredentials`	Returns a dictionary containing the credentials for all available protection spaces.

`credentialsForProtectionSpace:`	Returns a dictionary containing the credentials for the specified protection space.
`defaultCredentialForProtectionSpace:`	Returns the default credential for the specified protectionSpace.
`removeCredential:forProtectionSpace:`	Removes a specified credential from the credential storage for the specified protection space.
`setCredential:forProtectionSpace:`	Adds credential to the credential storage for the specified protectionSpace.
`setDefaultCredential:forProtectionSpace:`	Sets the default credential for a specified protection space.

NSURLDownload (New)

Complete reference information is available in the `NSURLDownload` reference.

Instance Methods

`cancel`	Cancels the receiver's download and deletes the downloaded file.
`initWithRequest:delegate:`	Returns an initialized URL download for a URL request and begins to download the data for the request.
`request`	Returns the request that initiated the receiver's download.
`setDestination:allowOverwrite:`	Sets the destination path of the downloaded file.

Delegate Methods

`download:decideDestinationWithSuggestedFilename:`	The delegate receives this message when download has determined a suggested filename for the downloaded file.
`download:didCancelAuthenticationChallenge:`	Sent if an authentication challenge is canceled due to the protocol implementation encountering an error.
`download:didCreateDestination:`	Sent when the destination file is created.
`download:didFailWithError:`	Sent if the download fails or if an I/O error occurs when the file is written to disk.

`download:didReceiveAuthenticationChallenge:`	Sent when the URL download must authenticate a challenge in order to download the request.
`download:didReceiveDataOfLength:`	Sent as a download object receives data incrementally.
`download:didReceiveResponse:`	Sent when a download object has received sufficient load data to construct the NSURLResponse object for the download.
`download:shouldDecodeSourceDataOfMIMEType:`	Sent when a download object determines that the downloaded file is encoded to inquire whether the file should be automatically decoded.
`download:willSendRequest:redirectResponse:`	Sent when the download object determines that it must change URLs in order to continue loading a request.
`downloadDidBegin:`	Sent immediately after a download object begins a download.
`downloadDidFinish:`	Sent when a download object has completed downloading successfully and has written its results to disk.

NSURLProtectionSpace (New)

Complete reference information is available in the `NSURLProtectionSpace` reference.

Instance Methods

`authenticationMethod`	Returns the authentication method used by the receiver.
`host`	Returns the receiver's host.
`initWithHost:port:protocol:realm:authenticationMethod:`	Initializes a protection space object.
`initWithProxyHost:port:type:realm:authenticationMethod:`	Initializes a protection space object representing a proxy server.
`isProxy`	Returns whether the receiver represents a proxy server.
`port`	Returns the receiver's port.
`protocol`	Returns the receiver's protocol.
`proxyType`	Returns the receiver's proxy type.

`realm`	Returns the receiver's authentication realm
`receivesCredentialSecurely`	Returns whether the credentials for the protection space can be sent securely.

NSURLProtocol (New)

Complete reference information is available in the `NSURLProtocol` reference.

Class Methods

`canInitWithRequest:`	Returns whether the protocol subclass can handle the specified request.
`canonicalRequestForRequest:`	Returns a canonical version of the specified request.
`propertyForKey:inRequest:`	Returns the property associated with the specified key in the specified request.
`registerClass:`	Attempts to register a subclass of NSURLProtocol, making it visible to the URL loading system.
`requestIsCacheEquivalent:toRequest:`	Returns whether two requests are equivalent for cache purposes.
`setProperty:forKey:inRequest:`	Sets the property associated with the specified key in the specified request.
`unregisterClass:`	Unregisters the specified subclass of NSURLProtocol.

Instance Methods

`cachedResponse`	Returns the receiver's cached response.
`client`	Returns the object the receiver uses to communicate with the URL loading system.
`initWithRequest:cachedResponse:client:`	Initializes an NSURLProtocol object.
`request`	Returns the receiver's request.
`startLoading`	Starts protocol-specific loading of the request.
`stopLoading`	Stops protocol-specific loading of the request.

NSURLRequest (New)

Complete reference information is available in the `NSURLRequest` reference.

Class Methods

`requestWithURL:`	Creates and returns a URL request for a specified URL with default cache policy and timeout value.
`requestWithURL:cachePolicy:timeoutInterval:`	Creates and returns an initialized URL request with specified values.

Instance Methods

`allHTTPHeaderFields`	Returns a dictionary containing all the receiver's HTTP header fields.
`cachePolicy`	Returns the receiver's cache policy.
`HTTPBody`	Returns the receiver's HTTP body data.
`HTTPMethod`	Returns the receiver's HTTP request method.
`HTTPShouldHandleCookies`	Returns whether the default cookie handling will be used for this request.
`initWithURL:`	Returns a URL request for a specified URL with default cache policy and timeout value.
`initWithURL:cachePolicy:timeoutInterval:`	Returns an initialized URL request with specified values.
`mainDocumentURL`	Returns the main document URL associated with the request.
`timeoutInterval`	Returns the receiver's timeout interval, in seconds.
`URL`	Returns the request's URL.
`valueForHTTPHeaderField:`	Returns the value of the specified HTTP header field.

NSURLResponse (New)

Complete reference information is available in the `NSURLResponse` reference.

Instance Methods

`expectedContentLength`	Returns the receiver's expected content length
`initWithURL:MIMEType:expectedContentLength:textEncodingName:`	Returns an initialized NSURLResponse object with the URL, MIME type, length, and text encoding set to given values.

`MIMEType`	Returns the receiver's MIME type.
`suggestedFilename`	Returns a suggested filename for the response data.
`textEncodingName`	Returns the name of the receiver's text encoding provided by the response's originating source.
`URL`	Returns the receiver's URL.

NSUserDefaults

Complete reference information is available in the `NSUserDefaults` reference.

Instance Methods

`objectIsForcedForKey:`	Returns a Boolean value indicating whether the specified key is managed by an administrator.
`objectIsForcedForKey:inDomain:`	Returns a Boolean value indicating whether the key in the specified domain is managed by an administrator.

Protocols

All of the protocols with new symbols are listed alphabetically, with their new methods described.

NSURLAuthenticationChallengeSender (New)

Complete reference information is available in the `NSURLAuthenticationChallengeSender` reference.

Instance Methods

`cancelAuthenticationChallenge:`	Cancels a given authentication challenge.
`continueWithoutCredentialFor-AuthenticationChallenge:`	Attempt to continue downloading a request without providing a credential for a given challenge.
`useCredential:forAuthenticationChallenge:`	Attempt to use a given credential for a given authentication challenge.

NSURLProtocolClient (New)

Complete reference information is available in the `NSURLProtocolClient` reference.

Instance Methods

`URLProtocol:cachedResponseIsValid:`	Sent to indicate to the URL loading system that a cached response is valid.
`URLProtocol:didCancelAuthenticationChallenge:`	Sent to indicate to the URL loading system that an authentication challenge has been canceled.
`URLProtocol:didFailWithError:`	Sent when the load request fails due to an error.
`URLProtocol:didLoadData:`	An NSURLProtocol subclass instance, protocol, sends this message to [protocol client] as it loads data.
`URLProtocol:didReceiveAuthenticationChallenge:`	Sent to indicate to the URL loading system that an authentication challenge has been received.
`URLProtocol:didReceiveResponse:cacheStoragePolicy:`	Sent to indicate to the URL loading system that the protocol implementation has created a response object for the request.
`URLProtocol:wasRedirectedToRequest: redirectResponse:`	Sent to indicate to the URL loading system that the protocol implementation has been redirected.
`URLProtocolDidFinishLoading:`	Sent to indicate to the URL loading system that the protocol implementation has finished loading.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

NSAppleScript.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSAppleScriptErrorAppName`	An NSString that specifies the name of the application that generated the error.
`NSAppleScriptErrorBriefMessage`	An NSString that provides a brief description of the error.
`NSAppleScriptErrorMessage`	An NSString that supplies a detailed description of the error condition.
`NSAppleScriptErrorNumber`	An NSNumber that specifies the error number.
`NSAppleScriptErrorRange`	An NSValue that specifies a range.

NSBundle.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSLocalizedStringWithDefaultValue`	Returns a localized version of a string.

NSError.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSLocalizedDescriptionKey`	The corresponding value is a localized string representation of the error that, if present, will be returned by localizedDescription.
`NSMachErrorDomain`	Mach errors
`NSOSStatusErrorDomain`	Mac OS 9/Carbon errors
`NSPOSIXErrorDomain`	POSIX/BSD errors

NSFileManager.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSFileAppendOnly	The key in a file attribute dictionary whose value indicates whether the file is read-only.
NSFileCreationDate	The key in a file attribute dictionary whose value indicates the file's creation date.
NSFileGroupOwnerAccountID	The key in a file attribute dictionary whose value indicates the file's group ID.
NSFileImmutable	The key in a file attribute dictionary whose value indicates whether the file is mutable.
NSFileOwnerAccountID	The key in a file attribute dictionary whose value indicates the file's owner's account ID.

NSHTTPCookie.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSHTTPCookieComment	An NSString object containing the comment for the cookie.
NSHTTPCookieCommentURL	An NSURL object or NSString object containing the comment URL for the cookie.
NSHTTPCookieDiscard	An NSString object stating whether the cookie should be discarded at the end of the session.
NSHTTPCookieDomain	An NSString object containing the domain for the cookie.
NSHTTPCookieExpires	An NSDate object or NSString object specifying the expiration date for the cookie.
NSHTTPCookieMaximumAge	An NSString object containing an integer value stating how long in seconds the cookie should be kept, at most.
NSHTTPCookieName	An NSString object containing the name of the cookie. This field is required.
NSHTTPCookieOriginURL	An NSURL or NSString object containing the URL that set this cookie.
NSHTTPCookiePath	An NSString object containing the path for the cookie.
NSHTTPCookiePort	An NSString object containing comma-separated integer values specifying the ports for the cookie.
NSHTTPCookieSecure	An NSString object stating whether the cookie should be transmitted only over secure channels.

NSHTTPCookieValue	An NSString object containing the value of the cookie.
NSHTTPCookieVersion	An NSString object that specifies the version of the cookie.

NSHTTPCookieStorage.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSHTTPCookieAcceptPolicy	NSHTTPCookieAcceptPolicy specifies the cookie acceptance policies implemented by the NSHTTPCookieStorage class.
NSHTTPCookieAcceptPolicyAlways	Accept all cookies. This is the default cookie accept policy.
NSHTTPCookieAcceptPolicyNever	Reject all cookies.
NSHTTPCookieAcceptPolicyOnlyFromMainDocumentDomain	Accept cookies only from the main document domain.
NSHTTPCookieManagerAcceptPolicyChangedNotification	This notification is posted when the acceptance policy of the NSHTTPCookieStorage instance has changed.
NSHTTPCookieManagerCookiesChangedNotification	This notification is posted when the cookies stored in the NSHTTPCookieStorage instance have changed.

NSInvocation.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSObjCBoolType	The BOOL type.

NSJavaSetup.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSJavaBundleCleanup`	This function has been deprecated.
`NSJavaBundleSetup`	This function has been deprecated.
`NSJavaClassesForBundle`	Loads the Java classes located in the specified bundle.
`NSJavaClassesFromPath`	Loads the Java classes located at the specified path.
`NSJavaObjectNamedInPath`	Creates an instance of the named class using the class loader previously specified at the given path.
`NSJavaSetup`	Loads the Java virtual machine with specified parameters.
`NSJavaSetupVirtualMachine`	Sets up the Java virtual machine.

NSKeyedArchiver.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSInvalidArchiveOperationException`	The name of the exception raised by NSKeyedArchiver if there is a problem creating an archive.
`NSInvalidUnarchiveOperationException`	The name of the exception raised by NSKeyedArchiver if there is a problem extracting an archive.

NSNetServices.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSNetServicesActivityInProgress`	The net service cannot process the request at this time. No additional information about the network state is known.

NSNetServicesBadArgumentError	An invalid argument was used when creating the NSNetService object.
NSNetServicesCancelledError	The client canceled the action.
NSNetServicesCollisionError	The service could not be published because the name is already in use. The name could be in use locally or on another system.
NSNetServicesError	These constants identify errors that can occur when accessing net services.
NSNetServicesErrorCode	This key identifies the error that occurred during the most recent operation.
NSNetServicesErrorDomain	This key identifies the originator of the error, which is either the NSNetService object or the mach network layer. For most errors, you should not need the value provided by this key.
NSNetServicesInvalidError	The net service was improperly configured.
NSNetServicesNotFoundError	The service could not be found on the network.
NSNetServicesUnknownError	An unknown error occurred.

NSObjCRuntime.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSFoundationVersionNumber10_1	Foundation version released in Mac OS X version 10.1.

NSPathUtilities.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

NSDocumentDirectory	Document directory.

NSPropertyList.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSPropertyListBinaryFormat_v1_0`	Specifies the binary property list format.
`NSPropertyListFormat`	These constants are used to specify a property list serialization format.
`NSPropertyListImmutable`	Causes the returned property list to contain immutable objects.
`NSPropertyListMutabilityOptions`	These constants specify mutability options in property lists.
`NSPropertyListMutableContainers`	Causes the returned property list to have mutable containers but immutable leaves.
`NSPropertyListMutableContainersAndLeaves`	Causes the returned property list to have mutable containers and leaves.
`NSPropertyListOpenStepFormat`	Specifies the old-style ASCII property list format inherited from the OpenStep APIs.
`NSPropertyListXMLFormat_v1_0`	Specifies the XML property list format.

NSURLCache.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLCacheStorageAllowed`	Specifies that storage in NSURLCache is allowed without restriction.
`NSURLCacheStorageAllowedInMemoryOnly`	Specifies that storage in NSURLCache is allowed; however storage should be restricted to memory only.
`NSURLCacheStorageNotAllowed`	Specifies that storage in NSURLCache is not allowed in any fashion, either in memory or on disk.
`NSURLCacheStoragePolicy`	These constants specify the caching strategy used by an NSCachedURLResponse object.

NSURLCredential.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLCredentialPersistence`	These constants specify how long the credential will be kept.
`NSURLCredentialPersistenceForSession`	Credential will be stored only for this session.
`NSURLCredentialPersistenceNone`	Credential won't be stored.
`NSURLCredentialPersistencePermanent`	Credential will be stored in the user's keychain and shared with other applications.

NSURLCredentialStorage.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLCredentialStorageChangedNotification`	This notification is posted when the set of stored credentials changes.

NSURLError.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSErrorFailingURLStringKey`	The corresponding value is the URL that caused the error. This key is only present in the NSURLErrorDomain.
`NSURLErrorBadServerResponse`	Returned when the URL Loading system receives bad data from the server.
`NSURLErrorBadURL`	Returned when a URL is sufficiently malformed that a URL request cannot be initiated
`NSURLErrorCancelled`	Returned when an asynchronous load is canceled.

NSURLErrorCannotCloseFile	Returned when NSURLDownload was unable to close the downloaded file on disk.
NSURLErrorCannotConnectToHost	Returned when an attempt to connect to a host has failed.
NSURLErrorCannotCreateFile	Returned when NSURLDownload object was unable to create the downloaded file on disk due to a I/O failure.
NSURLErrorCannotFindHost	Returned when the host name for a URL cannot be resolved.
NSURLErrorCannotLoadFromNetwork	Returned when a specific request to load an item only from the cache cannot be satisfied.
NSURLErrorCannotMoveFile	Returned when NSURLDownload was unable to move a downloaded file on disk.
NSURLErrorCannotOpenFile	Returned when NSURLDownload was unable to open the downloaded file on disk.
NSURLErrorCannotRemoveFile	Returned when NSURLDownload was unable to remove a downloaded file from disk.
NSURLErrorCannotWriteToFile	Returned when NSURLDownload was unable to write to the downloaded file on disk.
NSURLErrorDNSLookupFailed	See NSURLErrorCannotFindHost
NSURLErrorDomain	URL loading system errors
NSURLErrorDownloadDecodingFailedMidStream	Returned when NSURLDownload failed to decode an encoded file during the download.
NSURLErrorDownloadDecodingFailedToComplete	Returned when NSURLDownload failed to decode an encoded file after downloading.
NSURLErrorFileDoesNotExist	Returned when a file does not exist.
NSURLErrorFileIsDirectory	Returned when a request for an FTP file results in the server responding that the file is not a plain file, but a directory.
NSURLErrorHTTPTooManyRedirects	Returned when a redirect loop is detected or when the threshold for number of allowable redirects has been exceeded (currently 16).
NSURLErrorNetworkConnectionLost	Returned when a client or server connection is severed in the middle of an in-progress load.
NSURLErrorNoPermissionsToReadFile	Returned when a resource cannot be read due to insufficient permissions.

NSURLErrorNotConnectedToInternet	Returned when a network resource was requested, but an internet connection is not established and cannot be established automatically, either through a lack of connectivity, or by the user's choice not to make a network connection automatically.
NSURLErrorRedirectToNonExistentLocation	Returned when a redirect is specified by way of server response code, but the server does not accompany this code with a redirect URL.
NSURLErrorResourceUnavailable	Returned when a requested resource cannot be retrieved.
NSURLErrorSecureConnectionFailed	Returned when an attempt to establish a secure connection fails for reasons which cannot be expressed more specifically.
NSURLErrorServerCertificateHasBadDate	Returned when a server certificate has a date which indicates it has expired, or is not yet valid.
NSURLErrorServerCertificateHasUnknownRoot	Returned when a server certificate is not signed by any root server.
NSURLErrorServerCertificateUntrusted	Returned when a server certificate is signed by a root server which is not trusted.
NSURLErrorTimedOut	Returned when an asynchronous operation times out.
NSURLErrorUnknown	Returned when the URL Loading system encounters an error that it cannot interpret.
NSURLErrorUnsupportedURL	Returned when a properly formed URL cannot be handled by the framework.
NSURLErrorUserAuthenticationRequired	Returned when authentication is required to access a resource.
NSURLErrorUserCancelledAuthentication	Returned when an asynchronous request for authentication is cancelled by the user.
NSURLErrorZeroByteResource	Returned when a server reports that a URL has a non-zero content length, but terminates the network connection “gracefully” without sending any data.

NSURLHandle.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFTPPropertyActiveTransferModeKey`	Key for retrieving whether in active transfer mode, returned as a boolean wrapped in an NSNumber object.
`NSFTPPropertyFileOffsetKey`	Key for retrieving the file offset, returned as an NSNumber object. The default value for this key is zero.
`NSFTPPropertyUserLoginKey`	Key for the user login, returned as an NSString object.
`NSFTPPropertyUserPasswordKey`	Key for the user password, returned as an NSString object.
`NSHTTPPropertyHTTPProxy`	Key for retrieving the NSDictionary object containing proxy information to use in place of proxy identified in SystemConfiguration.framework.

NSURLProtectionSpace.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLAuthenticationMethodDefault`	Use the default authentication method for a protocol.
`NSURLAuthenticationMethodHTMLForm`	Use HTML form authentication for this protection space.
`NSURLAuthenticationMethodHTTPBasic`	Use HTTP basic authentication for this protection space.
`NSURLAuthenticationMethodHTTPDigest`	Use HTTP digest authentication for this protection space.
`NSURLProtectionSpaceFTPProxy`	The proxy type for FTP proxies.
`NSURLProtectionSpaceHTTPProxy`	The proxy type for HTTP proxies.
`NSURLProtectionSpaceHTTPSProxy`	The proxy type for HTTPS proxies.
`NSURLProtectionSpaceSOCKSProxy`	The proxy type for SOCKS proxies.

NSURLRequest.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLRequestCachePolicy`	These constants are used to specify interaction with the cached responses.
`NSURLRequestReloadIgnoringCacheData`	Replaced by NSURLRequestReloadIgnoringLocalCacheData.
`NSURLRequestReturnCacheDataDontLoad`	Specifies that the existing cache data should be used to satisfy a request, regardless of its age or expiration date.
`NSURLRequestReturnCacheDataElseLoad`	Specifies that the existing cached data should be used to satisfy the request, regardless of its age or expiration date. If there is no existing data in the cache corresponding the request, the data is loaded from the originating source.
`NSURLRequestUseProtocolCachePolicy`	Specifies that the caching logic defined in the protocol implementation, if any, is used for a particular URL load request. This is the default policy for URL load requests.

NSURLResponse.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSURLResponseUnknownLength`	Returned when the response length cannot be determined in advance of receiving the data from the server. For example, NSURLResponseUnknownLength is returned when the server HTTP response does not include a Content-Length header.

10.1 Symbol Changes

This article lists the symbols added to `Foundation.framework` in Mac OS X v10.1.

Classes

All of the classes with new symbols are listed alphabetically, with their new class, instance, and delegate methods described.

NSDictionary

Complete reference information is available in the `NSDictionary` reference.

Instance Methods

`fileExtensionHidden`	Returns the value for the NSFileExtensionHidden key.
`fileHFSCreatorCode`	Returns the value for the NSFileHFSCreatorCode key.
`fileHFSTypeCode`	Returns the value for the NSFileHFSTypeCode key.

NSFileManager

Complete reference information is available in the `NSFileManager` reference.

Instance Methods

`displayNameAtPath:`	Returns the name of the file or directory at a given path in a localized form appropriate for presentation to the user.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

NSFileManager.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFileExtensionHidden`	The key in a file attribute dictionary whose value indicates whether the file's extension is hidden.
`NSFileHFSCreatorCode`	The key in a file attribute dictionary whose value indicates the file's HFS creator code.
`NSFileHFSTypeCode`	The key in a file attribute dictionary whose value indicates the file's HFS type code.
`NSFoundationVersionWithFileManagerResourceForkSupport`	The version of the Foundation framework in which NSFileManager first supported resource forks.

NSObjCRuntime.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`NSFoundationVersionNumber`	The version of the Foundation framework in the current environment.
`NSFoundationVersionNumber10_0`	Foundation version released in Mac OS X version 10.0.

Document Revision History

This table describes the changes to *Foundation Reference Update*.

Date	Notes
2007-07-18	Updated with the symbols added to the Foundation framework in Mac OS X v10.5.
2006-05-23	Organized delegate methods and informal protocol methods.
2005-09-08	Added cross-reference to complete Foundation reference to table of contents.
2005-06-04	Made minor editorial revisions throughout.
2005-04-29	New document that summarizes the symbols added to the Foundation framework in Mac OS X v10.4.