Core Data 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

Introduction to Core Data Reference Update 5

Organization of This Document 5
See Also 5

10.4 - 10.5 Symbol Changes 7

Classes 7
- NSAtomicStore (New) 7
- NSAtomicStoreCacheNode (New) 8
- NSAttributeDescription 8
- NSEntityDescription 8
- NSEntityMapping (New) 9
- NSEntityMigrationPolicy (New) 10
- NSFetchRequest 10
- NSFetchRequestExpression (New) 11
- NSManagedObject 11
- NSManagedObjectContext 12
- NSManagedObjectModel 12
- NSMappingModel (New) 13
- NSMigrationManager (New) 13
- NSPersistentStore (New) 15
- NSPersistentStoreCoordinator 16
- NSPropertyDescription 16
- NSPropertyMapping (New) 16
- NSRelationshipDescription 17

C Symbols 17
- CoreDataErrors.h 17
- NSEntityMapping.h 18
- NSEntityMigrationPolicy.h 19
- NSFetchRequest.h 19
- NSFetchRequestExpression.h 19
- NSManagedObjectContext.h 20
- NSPersistentStoreCoordinator.h 20

10.3 - 10.4 Symbol Changes 21

Classes 21
- NSAttributeDescription (New) 21
- NSEntityDescription (New) 21
- NSFetchedPropertyDescription (New) 22
- NSFetchRequest (New) 23

NSManagedObject (New) 23
NSManagedObjectContext (New) 25
NSManagedObjectID (New) 28
NSManagedObjectModel (New) 28
NSPersistentStoreCoordinator (New) 29
NSPropertyDescription (New) 30
NSRelationshipDescription (New) 31
C Symbols 31
CoreDataDefines.h 32
CoreDataErrors.h 32
NSAttributeDescription.h 34
NSManagedObjectContext.h 35
NSPersistentStoreCoordinator.h 36
NSRelationshipDescription.h 36

Document Revision History 39

Introduction to Core Data Reference Update

This document summarizes the symbols that have been added to the Core Data 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 *Core Data Framework Reference*.

10.4 - 10.5 Symbol Changes

This document lists the symbols in CoreData that are new between Mac OS X v10.4 and 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.

NSAtomicStore (New)

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

Instance Methods

`addCacheNodes:`	Registers a set of cache nodes with the receiver.
`cacheNodeForObjectID:`	Returns the cache node for a given managed object ID.
`cacheNodes`	Returns the set of cache nodes registered with the receiver.
`initWithPersistentStoreCoordinator:configurationName:URL:options:`	Returns an atomic store, initialized with the given arguments.
`load:`	Loads the cache nodes for the receiver.
`newCacheNodeForManagedObject:`	Returns a new cache node for a given managed object.
`newReferenceObjectForManagedObject:`	Returns a new reference object for a given managed object.
`objectIDForEntity:referenceObject:`	Returns a managed object ID from the reference data for a specified entity.
`referenceObjectForObjectID:`	Returns the reference object for a given managed object ID.
`save:`	Saves the cache nodes.
`updateCacheNode:fromManagedObject:`	Updates the given cache node using the values in a given managed object.
`willRemoveCacheNodes:`	Method invoked before the store removes the given collection of cache nodes.

NSAtomicStoreCacheNode (New)

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

Instance Methods

`initWithObjectID:`	Returns a cache node for the given managed object ID.
`objectID`	Returns the managed object ID for the receiver.
`propertyCache`	Returns the property cache dictionary for the receiver.
`setPropertyCache:`	Sets the property cache dictionary for the receiver.
`setValue:forKey:`	Sets the value for the given key.
`valueForKey:`	Returns the value for a given key.

NSAttributeDescription

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

Instance Methods

`setAttributeValueClassName:`	Sets the name of the class used to represent the receiver.
`setValueTransformerName:`	Sets the name of the transformer to use to transform the attribute value.
`valueTransformerName`	Returns the name of the transformer used to transform the attribute value.
`versionHash`	Returns the version hash for the receiver.

NSEntityDescription

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

Instance Methods

`isKindOfEntity:`	Returns a Boolean value that indicates whether the receiver is a subentity of another given entity.
`setVersionHashModifier:`	Sets the version hash modifier for the receiver.
`versionHash`	Returns the version hash for the receiver.

`versionHashModifier`	Returns the version hash modifier for the receiver.

NSEntityMapping (New)

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

Instance Methods

`attributeMappings`	Returns the array of attribute mappings for the receiver.
`destinationEntityName`	Returns the destination entity name for the receiver.
`destinationEntityVersionHash`	Returns the version hash for the destination entity for the receiver.
`entityMigrationPolicyClassName`	Returns the class name of the migration policy for the receiver.
`mappingType`	Returns the mapping type for the receiver.
`name`	Returns the name of the receiver.
`relationshipMappings`	Returns the array of relationship mappings for the receiver.
`setAttributeMappings:`	Sets the array of attribute mappings for the receiver.
`setDestinationEntityName:`	Sets the destination entity name for the receiver.
`setDestinationEntityVersionHash:`	Sets the version hash for the destination entity for the receiver.
`setEntityMigrationPolicyClassName:`	Sets the class name of the migration policy for the receiver.
`setMappingType:`	Sets the mapping type for the receiver.
`setName:`	Sets the name of the receiver.
`setRelationshipMappings:`	Sets the array of relationship mappings for the receiver.
`setSourceEntityName:`	Sets the source entity name for the receiver.
`setSourceEntityVersionHash:`	Sets the version hash for the source entity for the receiver.
`setSourceExpression:`	Sets the source expression for the receiver.
`setUserInfo:`	Sets the user info dictionary for the receiver.
`sourceEntityName`	Returns the source entity name for the receiver.
`sourceEntityVersionHash`	Returns the version hash for the source entity for the receiver.

`sourceExpression`	Returns the source expression for the receiver.
`userInfo`	Returns the user info dictionary for the receiver.

NSEntityMigrationPolicy (New)

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

Instance Methods

`beginEntityMapping:manager:error:`	Invoked by the migration manager at the start of a given entity mapping.
`createDestinationInstancesForSourceInstance:entityMapping:manager:error:`	Creates the destination instance(s) for a given source instance.
`createRelationshipsForDestinationInstance:entityMapping:manager:error:`	Constructs the relationships between the newly-created destination instances.
`endEntityMapping:manager:error:`	Invoked by the migration manager at the end of a given entity mapping.
`endInstanceCreationForEntityMapping:manager:error:`	Indicates the end of the creation stage for the specified entity mapping, and the precursor to the next migration stage.
`endRelationshipCreationForEntityMapping:manager:error:`	Indicates the end of the relationship creation stage for the specified entity mapping.
`performCustomValidationForEntityMapping:manager:error:`	Invoked during the validation stage of the entity migration policy, providing the option of performing custom validation on migrated objects.

NSFetchRequest

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

Instance Methods

`includesPropertyValues`	Returns a Boolean value that indicates whether, when the fetch is executed, property data is obtained from the persistent store.

`includesSubentities`	Returns a Boolean value that indicates whether the receiver includes subentities in the results.
`relationshipKeyPathsForPrefetching`	Returns the array of relationship keypaths to prefetch along with the entity for the request.
`resultType`	Returns the result type of the receiver.
`returnsObjectsAsFaults`	Returns a Boolean value that indicates whether the objects resulting from a fetch using the receiver are faults.
`setIncludesPropertyValues:`	Sets if, when the fetch is executed, property data is obtained from the persistent store.
`setIncludesSubentities:`	Sets whether the receiver includes subentities.
`setRelationshipKeyPathsForPrefetching:`	Sets an array of relationship keypaths to prefetch along with the entity for the request.
`setResultType:`	Sets the result type of the receiver.
`setReturnsObjectsAsFaults:`	Sets whether the objects resulting from a fetch request are faults.

NSFetchRequestExpression (New)

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

Class Methods

`expressionForFetch:context:countOnly:`	Returns an expression which will evaluate to the result of executing a fetch request on a context.

Instance Methods

`contextExpression`	Returns the expression for the receiver's managed object context.
`isCountOnlyRequest`	Returns a Boolean value that indicates whether the receiver represents a count-only fetch request.
`requestExpression`	Returns the expression for the receiver's fetch request.

NSManagedObject

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

Instance Methods

`hasFaultForRelationshipNamed:`	Returns a Boolean value that indicates whether the relationship for a given key is a fault.
`willTurnIntoFault`	Invoked automatically by the Core Data framework before receiver is converted to a fault.

NSManagedObjectContext

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

Instance Methods

`countForFetchRequest:error:`	Returns the number of objects a given fetch request would have returned if it had been passed to executeFetchRequest:error:.
`mergeChangesFromContextDidSaveNotification:`	Merges the changes specified in a given notification.
`obtainPermanentIDsForObjects:error:`	Converts to permanent IDs the object IDs of the objects in a given array.

NSManagedObjectModel

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

Class Methods

`mergedModelFromBundles:forStoreMetadata:`	Returns a merged model from a specified array for the version information in provided metadata.
`modelByMergingModels:forStoreMetadata:`	Returns, for the version information in given metadata, a model merged from a given array of models.

Instance Methods

`entityVersionHashesByName`	Returns a dictionary of the version hashes for the entities in the receiver.
`fetchRequestTemplatesByName`	Returns a dictionary of the receiver's fetch request templates.

`isConfiguration:compatibleWithStoreMetadata:`	Returns a Boolean value that indicates whether a given configuration in the receiver is compatible with given metadata from a persistent store.
`setVersionIdentifiers:`	Sets the identifiers for the receiver.
`versionIdentifiers`	Returns the collection of developer-defined version identifiers for the receiver.

NSMappingModel (New)

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

Class Methods

`mappingModelFromBundles:forSourceModel: destinationModel:`	Returns the mapping model to translate data from the source to the destination model.

Instance Methods

`entityMappings`	Returns the collection of entity mappings for the receiver.
`entityMappingsByName`	Returns a dictionary of the entity mappings for the receiver.
`initWithContentsOfURL:`	Returns a mapping model initialized from a given URL.
`setEntityMappings:`	Sets the collection of entity mappings for the receiver

NSMigrationManager (New)

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

Instance Methods

`associateSourceInstance:withDestinationInstance: forEntityMapping:`	Associates a given source instance with an array of destination instances for a given property mapping.
`cancelMigrationWithError:`	Cancels the migration with a given error.
`currentEntityMapping`	Returns the entity mapping currently being processed.

`destinationContext`	Returns the managed object context the receiver uses for writing the destination persistent store.
`destinationEntityForEntityMapping:`	Returns the entity description for the destination entity of a given entity mapping.
`destinationInstancesForEntityMappingNamed: sourceInstances:`	Returns the managed object instances created in the destination store for a named entity mapping for a given array of source instances.
`destinationModel`	Returns the destination model for the receiver.
`initWithSourceModel:destinationModel:`	Initializes a migration manager instance with given source and destination models.
`mappingModel`	Returns the mapping model for the receiver.
`migrateStoreFromURL:type:options:withMappingModel: toDestinationURL:destinationType: destinationOptions:error:`	Migrates of the store at a given source URL to the store at a given destination URL, performing all of the mappings specified in a given mapping model.
`migrationProgress`	Returns a number from 0 to 1 that indicates the proportion of completeness of the migration.
`reset`	Resets the association tables for the migration.
`setUserInfo:`	Sets the user info for the receiver.
`sourceContext`	Returns the managed object context the receiver uses for reading the source persistent store.
`sourceEntityForEntityMapping:`	Returns the entity description for the source entity of a given entity mapping.
`sourceInstancesForEntityMappingNamed: destinationInstances:`	Returns the managed object instances in the source store used to create a given destination instance for a given property mapping.
`sourceModel`	Returns the source model for the receiver.
`userInfo`	Returns the user info for the receiver.

NSPersistentStore (New)

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

Class Methods

`metadataForPersistentStoreWithURL:error:`	Returns the metadata from the persistent store at the given URL.
`setMetadata:forPersistentStoreWithURL:error:`	Sets the metadata for the store at a given URL.

Instance Methods

`configurationName`	Returns the name of the managed object model configuration used to create the receiver.
`didAddToPersistentStoreCoordinator:`	Invoked after the receiver has been added to the persistent store coordinator.
`identifier`	Returns the unique identifier for the receiver.
`initWithPersistentStoreCoordinator: configurationName:URL:options:`	Returns a store initialized with the given arguments.
`isReadOnly`	Returns a Boolean value that indicates whether the receiver is read-only.
`metadata`	Returns the metadata for the receiver.
`options`	Returns the options with which the receiver was created.
`persistentStoreCoordinator`	Returns the persistent store coordinator which loaded the receiver.
`setIdentifier:`	Sets the unique identifier for the receiver.
`setMetadata:`	Sets the metadata for the receiver.
`setReadOnly:`	Sets whether the receiver is read-only.
`setURL:`	Sets the URL for the receiver.
`type`	Returns the type string of the receiver.
`willRemoveFromPersistentStoreCoordinator:`	Invoked before the receiver is removed from the persistent store coordinator.

NSPersistentStoreCoordinator

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

Class Methods

`metadataForPersistentStoreOfType:URL:error:`	Returns a dictionary containing the metadata stored in the persistent store at a given URL.
`registeredStoreTypes`	Returns a dictionary of the registered store types.
`registerStoreClass:forStoreType:`	Registers a given NSPersistentStore subclass for a given store type string.
`setMetadata:forPersistentStoreOfType:URL:error:`	Sets the metadata for a given store.

Instance Methods

`setURL:forPersistentStore:`	Sets the URL for a given persistent store.

NSPropertyDescription

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

Instance Methods

`isIndexed`	Returns a Boolean value that indicates whether the receiver is important for searching.
`setIndexed:`	Sets the optionality flag of the receiver.
`setVersionHashModifier:`	Sets the version hash modifier for the receiver.
`versionHash`	Returns the version hash for the receiver.
`versionHashModifier`	Returns the version hash modifier for the receiver.

NSPropertyMapping (New)

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

Instance Methods

`name`	Returns the name of the property in the destination entity for the receiver.
`setName:`	Sets the name of the property in the destination entity for the receiver.
`setUserInfo:`	Sets the user info for the receiver.
`setValueExpression:`	Sets the value expression for the receiver.
`userInfo`	Returns the user info for the receiver.
`valueExpression`	Returns the value expression for the receiver.

NSRelationshipDescription

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

Instance Methods

`versionHash`	Returns the version hash for the receiver.

C Symbols

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

CoreDataErrors.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.

`NSEntityMigrationPolicyError`	Error code to denote that migration failed during processing of an entity migration policy.
`NSMigrationCancelledError`	Error code to denote that migration failed due to manual cancellation.
`NSMigrationError`	Error code to denote a general migration error.

`NSMigrationManagerDestinationStoreError`	Error code to denote that migration failed due to a problem with the destination data store.
`NSMigrationManagerSourceStoreError`	Error code to denote that migration failed due to a problem with the source data store.
`NSMigrationMissingMappingModelError`	Error code to denote that migration failed due to a missing mapping model.
`NSMigrationMissingSourceModelError`	Error code to denote that migration failed due to a missing source data model.
`NSPersistentStoreIncompatibleVersionHashError`	Error code to denote that entity version hashes in the store are incompatible with the current managed object model.
`NSPersistentStoreOpenError`	Error code to denote an error occurred while attempting to open a persistent store.
`NSPersistentStoreOperationError`	Error code to denote that a persistent store operation failed.
`NSPersistentStoreTimeoutError`	Error code to denote that Core Data failed to connect to a persistent store within the time specified by NSPersistentStoreTimeoutOption.
`NSSQLiteError`	Error code to denote a general SQLite error.
`NSSQLiteErrorDomain`	Domain for SQLite errors.

NSEntityMapping.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.

`NSAddEntityMappingType`	Specifies that this is a new entity in the destination model.
`NSCopyEntityMappingType`	Specifies that source instances are migrated as-is.
`NSCustomEntityMappingType`	Specifies a custom mapping.
`NSEntityMappingType`	Data type used for constants that specify types of entity mapping.
`NSRemoveEntityMappingType`	Specifies that this entity is not present in the destination model.

`NSTransformEntityMappingType`	Specifies that entity exists in source and destination and is mapped.
`NSUndefinedEntityMappingType`	Specifies that the developer handles destination instance creation.

NSEntityMigrationPolicy.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.

`NSMigrationDestinationObjectKey`	Key for the destination object.
`NSMigrationEntityMappingKey`	Key for the entity mapping object.
`NSMigrationManagerKey`	Key for the migration manager.
`NSMigrationPropertyMappingKey`	Key for the property mapping object.
`NSMigrationSourceObjectKey`	Key for the source object.

NSFetchRequest.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.

`NSFetchRequestResultType`	Defines the type for the fetch request result type.
`NSManagedObjectIDResultType`	Specifies that the request returns managed object IDs.
`NSManagedObjectResultType`	Specifies that the request returns managed objects.

NSFetchRequestExpression.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.

`NSFetchRequestExpressionType`	Specifies the fetch request expression type.

NSManagedObjectContext.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.

`NSInvalidatedAllObjectsKey`	Key that specifies that all objects in the context have been invalidated.
`NSInvalidatedObjectsKey`	Key for the set of objects that were invalidated.
`NSRefreshedObjectsKey`	Key for the set of objects that were refreshed.

NSPersistentStoreCoordinator.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.

`NSIgnorePersistentStoreVersioningOption`	Key to ignore the built-in versioning provided by Core Data.
`NSMigratePersistentStoresAutomaticallyOption`	Key to automatically attempt to migrate versioned stores.
`NSPersistentStoreOSCompatibility`	Key to represent the earliest version of Mac OS X the persistent store supports.
`NSPersistentStoreTimeoutOption`	Options key that specifies the connection timeout for Core Data stores.
`NSSQLitePragmasOption`	Options key for a dictionary of SQLite pragma settings with pragma values indexed by pragma names as keys.
`NSStoreModelVersionHashesKey`	Key to represent the version hash information for the model used to create the store.
`NSStoreModelVersionIdentifiersKey`	Key to represent the version identifier for the model used to create the store.

10.3 - 10.4 Symbol Changes

This document lists the symbols in CoreData that are new between Mac OS X v10.3 and 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.

NSAttributeDescription (New)

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

Instance Methods

`attributeType`	Returns the type of the receiver.
`attributeValueClassName`	Returns the name of the class used to represent the receiver.
`defaultValue`	Returns the default value of the receiver.
`setAttributeType:`	Sets the type of the receiver.
`setDefaultValue:`	Sets the default value of the receiver.

NSEntityDescription (New)

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

Class Methods

`entityForName:inManagedObjectContext:`	Returns the entity with the specified name from the managed object model associated with the specified managed object context's persistent store coordinator.
`insertNewObjectForEntityForName:inManagedObjectContext:`	Creates, configures, and returns a new autoreleased instance of the class for the entity with a given name.

Instance Methods

`attributesByName`	Returns the attributes of the receiver in a dictionary, where the keys in the dictionary are the attribute names.
`isAbstract`	Returns a Boolean value that indicates whether the receiver represents an abstract entity.
`managedObjectClassName`	Returns the name of the class that represents the receiver's entity.
`managedObjectModel`	Returns the managed object model with which the receiver is associated.
`name`	Returns the entity name of the receiver.
`properties`	Returns an array containing the properties of the receiver.
`propertiesByName`	Returns a dictionary containing the properties of the receiver.
`relationshipsByName`	Returns the relationships of the receiver in a dictionary, where the keys in the dictionary are the relationship names.
`relationshipsWithDestinationEntity:`	Returns an array containing the relationships of the receiver where the entity description of the relationship is a given entity.
`setAbstract:`	Sets whether the receiver represents an abstract entity.
`setManagedObjectClassName:`	Sets the name of the class that represents the receiver's entity.
`setName:`	Sets the entity name of the receiver.
`setProperties:`	Sets the properties array of the receiver.
`setSubentities:`	Sets the subentities of the receiver.
`setUserInfo:`	Sets the user info dictionary of the receiver.
`subentities`	Returns an array containing the sub-entities of the receiver.
`subentitiesByName`	Returns the sub-entities of the receiver in a dictionary.
`superentity`	Returns the super-entity of the receiver.
`userInfo`	Returns the user info dictionary of the receiver.

NSFetchedPropertyDescription (New)

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

Instance Methods

`fetchRequest`	Returns the fetch request of the receiver.
`setFetchRequest:`	Sets the fetch request of the receiver.

NSFetchRequest (New)

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

Instance Methods

`affectedStores`	Returns an array containing the persistent stores specified for the receiver.
`entity`	Returns the entity specified for the receiver.
`fetchLimit`	Returns the fetch limit of the receiver.
`predicate`	Returns the predicate of the receiver.
`setAffectedStores:`	Sets the array of persistent stores that will be searched by the receiver.
`setEntity:`	Sets the entity of the receiver.
`setFetchLimit:`	Sets the fetch limit of the receiver.
`setPredicate:`	Sets the predicate of the receiver.
`setSortDescriptors:`	Sets the array of sort descriptors of the receiver.
`sortDescriptors`	Returns the sort descriptors of the receiver.

NSManagedObject (New)

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

Instance Methods

`awakeFromFetch`	Invoked automatically by the Core Data framework after the receiver has been fetched.
`awakeFromInsert`	Invoked automatically by the Core Data framework when the receiver is first inserted into a managed object context.

`changedValues`	Returns a dictionary containing the keys and (new) values of persistent properties that have been changed since last fetching or saving the receiver.
`committedValuesForKeys:`	Returns a dictionary of the last fetched or saved values of the receiver for the properties specified by the given keys.
`didAccessValueForKey:`	Provides support for key-value observing access notification.
`didChangeValueForKey:`	Provides support for key-value observing change notification.
`didChangeValueForKey:withSetMutation:usingObjects:`	Provides support for key-value observing change notifications for to-many relationships.
`didSave`	Invoked automatically by the Core Data framework after the receiver's managed object context completes a save operation.
`didTurnIntoFault`	Invoked automatically by the Core Data framework when the receiver is turned into a fault.
`entity`	Returns the entity description of the receiver.
`initWithEntity:insertIntoManagedObjectContext:`	Initializes the receiver and inserts it into the specified managed object context.
`isDeleted`	Returns a Boolean value that indicates whether the receiver will be deleted during the next save.
`isFault`	Returns a Boolean value that indicates whether the receiver is a fault.
`isInserted`	Returns a Boolean value that indicates whether the receiver has been inserted in a managed object context.
`isUpdated`	Returns a Boolean value that indicates whether the receiver has unsaved changes.
`managedObjectContext`	Returns the managed object context with which the receiver is registered.

`objectID`	Returns the object ID of the receiver.
`observationInfo`	Returns the observation info of the receiver.
`primitiveValueForKey:`	Returns from the receiver's private internal storage the value for the specified property.
`setObservationInfo:`	Sets the observation info of the receiver.
`setPrimitiveValue:forKey:`	Sets in the receiver's private internal storage the value of a given property.
`setValue:forKey:`	Sets the specified property of the receiver to the specified value.
`validateForDelete:`	Determines whether the receiver can be deleted in its current state.
`validateForInsert:`	Determines whether the receiver can be inserted in its current state.
`validateForUpdate:`	Determines whether the receiver's current state is valid.
`validateValue:forKey:error:`	Validates a property value for a given key.
`valueForKey:`	Returns the value for the property specified by key.
`willAccessValueForKey:`	Provides support for key-value observing access notification.
`willChangeValueForKey:`	Provides support for key-value observing change notification.
`willChangeValueForKey:withSetMutation: usingObjects:`	Provides support for key-value observing change notifications for to-many relationships.
`willSave`	Invoked automatically by the Core Data framework when the receiver's managed object context is saved.

NSManagedObjectContext (New)

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

Instance Methods

`assignObject:toPersistentStore:`	Specifies the store in which a newly-inserted object will be saved.
`deletedObjects`	Returns the set of objects that will be removed from their persistent store during the next save operation.
`deleteObject:`	Specifies an object that should be removed from its persistent store when changes are committed.
`detectConflictsForObject:`	Marks an object for conflict detection.
`executeFetchRequest:error:`	Returns an array of objects that meet the criteria specified by a given fetch request.
`hasChanges`	Returns a Boolean value that indicates whether the receiver has uncommitted changes.
`insertedObjects`	Returns the set of objects that have been inserted into the receiver but not yet saved in a persistent store.
`insertObject:`	Registers an object to be inserted in the receiver's persistent store the next time changes are saved.
`lock`	Attempts to acquire a lock on the receiver.
`mergePolicy`	Returns the merge policy of the receiver.
`objectRegisteredForID:`	Returns the object for a specified ID, if the object is registered with the receiver.
`objectWithID:`	Returns the object for a specified ID.
`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.
`persistentStoreCoordinator`	Returns the persistent store coordinator of the receiver.
`processPendingChanges`	Forces the receiver to process changes to the object graph.
`propagatesDeletesAtEndOfEvent`	Returns a Boolean that indicates whether the receiver propagates deletes at the end of the event in which a change was made.

`redo`	Sends an redo message to the receiver's undo manager, asking it to reverse the latest undo operation applied to objects in the object graph.
`refreshObject:mergeChanges:`	Updates the persistent properties of a managed object to use the latest values from the persistent store.
`registeredObjects`	Returns the set of objects registered with the receiver.
`reset`	Returns the receiver to its base state.
`retainsRegisteredObjects`	Returns a Boolean that indicates whether the receiver sends a retain message to objects upon registration.
`rollback`	Removes everything from the undo stack, discards all insertions and deletions, and restores updated objects to their last committed values.
`save:`	Attempts to commit unsaved changes to registered objects to their persistent store.
`setMergePolicy:`	Sets the merge policy of the receiver.
`setPersistentStoreCoordinator:`	Sets the persistent store coordinator of the receiver.
`setPropagatesDeletesAtEndOfEvent:`	Sets whether the context propagates deletes to related objects at the end of the event.
`setRetainsRegisteredObjects:`	Sets whether or not the receiver retains all registered objects, or only objects necessary for a pending save (those that are inserted, updated, deleted, or locked).
`setStalenessInterval:`	Sets the staleness interval of the receiver.
`setUndoManager:`	Sets the undo manager of the receiver.
`stalenessInterval`	Returns the staleness interval of the receiver.
`tryLock`	Attempts to acquire a lock.
`undo`	Sends an undo message to the receiver's undo manager, asking it to reverse the latest uncommitted changes applied to objects in the object graph.
`undoManager`	Returns the undo manager of the receiver.

`unlock`	Relinquishes a previously acquired lock.
`updatedObjects`	Returns the set of objects registered with the receiver that have uncommitted changes.

NSManagedObjectID (New)

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

Instance Methods

`entity`	Returns the entity description associated with the receiver.
`isTemporaryID`	Returns a Boolean value that indicates whether the receiver is temporary.
`persistentStore`	Returns the persistent store that contains the object whose ID is the receiver.
`URIRepresentation`	Returns a URI that provides an archiveable reference to the object which the receiver represents.

NSManagedObjectModel (New)

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

Class Methods

`mergedModelFromBundles:`	Returns a model created by merging all the models found in given bundles.
`modelByMergingModels:`	Creates a single model from an array of existing models.

Instance Methods

`configurations`	Returns all the available configuration names of the receiver.
`entities`	Returns the entities in the receiver.
`entitiesByName`	Returns the entities of the receiver in a dictionary.
`entitiesForConfiguration:`	Returns the entities of the receiver for a specified configuration.

`fetchRequestFromTemplateWithName:substitutionVariables:`	Returns a copy of the fetch request template with the variables substituted by values from the substitutions dictionary.
`fetchRequestTemplateForName:`	Returns the fetch request with a specified name.
`init`	
`initWithContentsOfURL:`	Initializes the receiver using the model file at the specified URL.
`localizationDictionary`	Returns the localization dictionary of the receiver.
`setEntities:`	Sets the entities array of the receiver.
`setEntities:forConfiguration:`	Associates the specified entities with the receiver using the given configuration name.
`setFetchRequestTemplate:forName:`	Associates the specified fetch request with the receiver using the given name.
`setLocalizationDictionary:`	Sets the localization dictionary of the receiver.

NSPersistentStoreCoordinator (New)

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

Class Methods

`metadataForPersistentStoreWithURL:error:`	Returns a dictionary that contains the metadata stored in the persistent store at the specified location.

Instance Methods

`addPersistentStoreWithType:configuration:URL:options:error:`	Adds a new persistent store of a specified type at a given location, and returns the new store.
`initWithManagedObjectModel:`	Initializes the receiver with a managed object model.
`lock`	Attempts to acquire a lock.
`managedObjectIDForURIRepresentation:`	Returns an object ID for the specified URI representation of an object ID if a matching store is available, or nil if a matching store cannot be found.
`managedObjectModel`	Returns the receiver's managed object model.

`metadataForPersistentStore:`	Returns a dictionary that contains the metadata currently stored or to-be-stored in a given persistent store.
`migratePersistentStore:toURL:options:withType:error:`	Moves a persistent store to a new location, changing the storage type if necessary.
`persistentStoreForURL:`	Returns the persistent store for the specified URL.
`persistentStores`	Returns an array of persistent stores associated with the receiver.
`removePersistentStore:error:`	Removes a given persistent store.
`setMetadata:forPersistentStore:`	Sets the metadata stored in the persistent store during the next save operation executed on it to metadata.
`tryLock`	Attempts to acquire a lock.
`unlock`	Relinquishes a previously acquired lock.
`URLForPersistentStore:`	Returns the URL for a given persistent store.

NSPropertyDescription (New)

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

Instance Methods

`entity`	Returns the entity description of the receiver.
`isOptional`	Returns a Boolean value that indicates whether the receiver is optional.
`isTransient`	Returns a Boolean value that indicates whether the receiver is transient.
`name`	Returns the name of the receiver.
`setName:`	Sets the name of the receiver.
`setOptional:`	Sets the optionality flag of the receiver.
`setTransient:`	Sets the transient flag of the receiver.
`setUserInfo:`	Sets the user info dictionary of the receiver.
`setValidationPredicates:withValidationWarnings:`	Sets the validation predicates and warnings of the receiver.

`userInfo`	Returns the user info dictionary of the receiver.
`validationPredicates`	Returns the validation predicates of the receiver.
`validationWarnings`	Returns the error strings associated with the receiver's validation predicates.

NSRelationshipDescription (New)

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

Instance Methods

`deleteRule`	Returns the delete rule of the receiver.
`destinationEntity`	Returns the entity description of the receiver's destination.
`inverseRelationship`	Returns the relationship that represents the inverse of the receiver.
`isToMany`	Returns a Boolean value that indicates whether the receiver represents a to-many relationship.
`maxCount`	Returns the maximum count of the receiver.
`minCount`	Returns the minimum count of the receiver.
`setDeleteRule:`	Sets the delete rule of the receiver.
`setDestinationEntity:`	Sets the entity description for the receiver's destination.
`setInverseRelationship:`	Sets the inverse relationship of the receiver.
`setMaxCount:`	Sets the maximum count of the receiver.
`setMinCount:`	Sets the minimum count of the receiver.

C Symbols

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

CoreDataDefines.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.

`NSCoreDataVersionNumber`	Specifies the version of Core Data available in the current process.

CoreDataErrors.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.

`NSAffectedObjectsErrorKey`	The key for objects prompting an error.
`NSAffectedStoresErrorKey`	The key for stores prompting an error.
`NSDetailedErrorsKey`	If multiple validation errors occur in one operation, they are collected in an array and added with this key to the "top-level error" of the operation.
`NSManagedObjectContextLockingError`	Error code to denote an inability to acquire a lock in a managed object context.
`NSManagedObjectExternalRelationshipError`	Error code to denote that an object being saved has a relationship containing an object from another store.
`NSManagedObjectMergeError`	Error code to denote that a merge policy failed—Core Data is unable to complete merging.
`NSManagedObjectReferentialIntegrityError`	Error code to denote an attempt to fire a fault pointing to an object that does not exist.
`NSManagedObjectValidationError`	Error code to denote a generic validation error.
`NSPersistentStoreCoordinatorLockingError`	Error code to denote an inability to acquire a lock in a persistent store.

`NSPersistentStoreIncompatibleSchemaError`	Error code to denote that a persistent store returned an error for a save operation.
`NSPersistentStoreIncompleteSaveError`	Error code to denote that one or more of the stores returned an error during a save operations.
`NSPersistentStoreInvalidTypeError`	Error code to denote an unknown persistent store type/format/version.
`NSPersistentStoreSaveError`	Error code to denote that a persistent store returned an error for a save operation.
`NSPersistentStoreTypeMismatchError`	Error code returned by a persistent store coordinator if a store is accessed that does not match the specified type.
`NSValidationDateTooLateError`	Error code to denote some date value is too late.
`NSValidationDateTooSoonError`	Error code to denote some date value is too soon.
`NSValidationInvalidDateError`	Error code to denote some date value fails to match date pattern.
`NSValidationKeyErrorKey`	Key for the key that failed to validate for a validation error.
`NSValidationMissingMandatoryPropertyError`	Error code for a non-optional property with a nil value.
`NSValidationMultipleErrorsError`	Error code to denote an error containing multiple validation errors.
`NSValidationNumberTooLargeError`	Error code to denote some numerical value is too large.
`NSValidationNumberTooSmallError`	Error code to denote some numerical value is too small.
`NSValidationObjectErrorKey`	Key for the object that failed to validate for a validation error.
`NSValidationPredicateErrorKey`	For predicate-based validation, key for the predicate for the condition that failed to validate.
`NSValidationRelationshipDeniedDeleteError`	Error code to denote some relationship with delete rule NSDeleteRuleDeny is non-empty.

`NSValidationRelationshipExceedsMaximumCountError`	Error code to denote a bounded to-many relationship with too many destination objects.
`NSValidationRelationshipLacksMinimumCountError`	Error code to denote a to-many relationship with too few destination objects.
`NSValidationStringPatternMatchingError`	Error code to denote some string value fails to match some pattern.
`NSValidationStringTooLongError`	Error code to denote some string value is too long.
`NSValidationStringTooShortError`	Error code to denote some string value is too short.
`NSValidationValueErrorKey`	If non-nil, the key for the value for the key that failed to validate for a validation error.

NSAttributeDescription.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.

`NSAttributeType`	Defines the possible types of NSAttributeType properties. These explicitly distinguish between bit sizes to ensure data store independence.
`NSBinaryDataAttributeType`	Specifies an NSData attribute.
`NSBooleanAttributeType`	Specifies a Boolean attribute.
`NSDateAttributeType`	Specifies an NSDate attribute.
`NSDecimalAttributeType`	Specifies an NSDecimalNumber attribute.
`NSDoubleAttributeType`	Specifies a double attribute.
`NSFloatAttributeType`	Specifies a float attribute.
`NSInteger16AttributeType`	Specifies a 16-bit signed integer attribute.
`NSInteger32AttributeType`	Specifies a 32-bit signed integer attribute.
`NSInteger64AttributeType`	Specifies a 64-bit signed integer attribute.
`NSStringAttributeType`	Specifies an NSString attribute.

NSUndefinedAttributeType	Specifies an undefined attribute type.

NSManagedObjectContext.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.

NSDeletedObjectsKey	Key for the set of objects that were marked for deletion during the previous event.
NSErrorMergePolicy	This policy causes a save to fail if there are any merge conflicts.
NSInsertedObjectsKey	Key for the set of objects that were inserted into the context.
NSManagedObjectContextDidSaveNotification	Posted whenever a managed object context completes a save operation.
NSManagedObjectContextObjectsDidChangeNotification	Posted when values of properties of objects contained in a managed object context are changed.
NSMergeByPropertyObjectTrumpMergePolicy	This policy merges conflicts between the persistent store's version of the object and the current in-memory version, giving priority to in-memory changes.
NSMergeByPropertyStoreTrumpMergePolicy	This policy merges conflicts between the persistent store's version of the object and the current in-memory version, giving priority to external changes.
NSOverwriteMergePolicy	This policy overwrites state in the persistent store for the changed objects in conflict.
NSRollbackMergePolicy	This policy discards in-memory state changes for objects in conflict.
NSUpdatedObjectsKey	Key for the set of objects that were updated.

NSPersistentStoreCoordinator.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.

`NSAddedPersistentStoresKey`	Key for the array of stores that were added.
`NSBinaryStoreType`	The binary store type.
`NSInMemoryStoreType`	The in-memory store type.
`NSPersistentStoreCoordinator-StoresDidChangeNotification`	Posted whenever persistent stores are added to or removed from a persistent store coordinator, or when store UUIDs change.
`NSReadOnlyPersistentStoreOption`	A flag that indicates whether a store is treated as read-only or not.
`NSRemovedPersistentStoresKey`	Key for the array of stores that were removed.
`NSSQLiteStoreType`	The SQLite database store type.
`NSStoreTypeKey`	The key in the metadata dictionary to identify the store type.
`NSStoreUUIDKey`	The key in the metadata dictionary to identify the store UUID.
`NSUUIDChangedPersistentStoresKey`	Key for the array of stores whose UUIDs changed.
`NSValidateXMLStoreOption`	A flag that indicates whether an XML file should be validated with the DTD while opening.
`NSXMLStoreType`	The XML store type.

NSRelationshipDescription.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.

`NSCascadeDeleteRule`	If the object is deleted, the destination object or objects of this relationship are also deleted.
`NSDeleteRule`	These constants define what happens to relationships when an object is deleted.
`NSDenyDeleteRule`	If the destination of this relationship is not nil, the delete creates a validation error.

`NSNoActionDeleteRule`	If the object is deleted, no modifications are made to objects at the destination of the relationship.
`NSNullifyDeleteRule`	If the object is deleted, back pointers from the objects to which it is related are nullified.

Document Revision History

This table describes the changes to *Core Data Reference Update*.

Date	Notes
2007-07-18	Updated for Mac OS X v10.5.
2006-03-08	Corrected minor typographical errors.
2005-09-08	Added cross-reference to complete Core Data reference to table of contents.
2005-07-07	Corrected minor formatting errors.
2005-04-29	Minor corrections to conform with style guidelines; added See Also section to Introduction.
	New document that summarizes the symbols added to the Core Data framework in Mac OS X v10.4.