# NSPersistentStoreCoordinator Class Reference

**Data Management**

**2009-05-01**

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



Simultaneously published in the United States and Canada.

# Contents

# NSPersistentStoreCoordinator Class Reference

| | |
|---|---|
| **Inherits from** | NSObject |
| **Conforms to** | NSLocking<br>NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/CoreData.framework |
| **Availability** | Available in Mac OS X v10.4 and later. |
| **Declared in** | NSPersistentStoreCoordinator.h |
| **Companion guides** | Core Data Programming Guide<br>Atomic Store Programming Topics<br>Core Data Spotlight Integration Programming Guide |
| **Related sample code** | AbstractTree<br>Core Data HTML Store<br>CoreRecipes<br>CustomAtomicStoreSubclass<br>Image Kit with Core Data |

## Overview

Instances of `NSPersistentStoreCoordinator` associate persistent stores (by type) with a model (or more accurately, a configuration of a model) and serve to mediate between the persistent store or stores and the managed object context or contexts. Instances of `NSManagedObjectContext` use a coordinator to save object graphs to persistent storage and to retrieve model information. A context without a coordinator is not fully functional as it cannot access a model except through a coordinator. The coordinator is designed to present a façade to the managed object contexts such that a group of persistent stores appears as an aggregate store. A managed object context can then create an object graph based on the union of all the data stores the coordinator covers.

Coordinators do the opposite of providing for concurrency—they serialize operations. If you want to use multiple threads for different write operations you use multiple coordinators. Note that if multiple threads work directly with a coordinator, they need to lock and unlock it explicitly.

Each coordinator (and thus container) may use different copies, and hence different versions, of a managed object model. This allows you to cleanly deal with file versioning.

The coordinator gives access to its underlying object stores. You can retrieve an object store when you first add one (using `addPersistentStoreWithType:configuration:URL:options:error:` (page 10)), or by using `persistentStoreForURL:` (page 16) or `persistentStores` (page 16). This allows you to to determine, for example, whether a store has already been added, or whether two objects come from the same store.

- You move a store from one location to another, or change the type of a store, using migratePersistentStore:toURL:options:withType:error: (page 15).
- You can set metadata for a given store using the persistent store coordinator (`setMetadata:forPersistentStore:` (page 17)).

For more details about these tasks, see Persistent Store Features in *Core Data Programming Guide*.

# Tasks

## Registered Store Types

`+ registeredStoreTypes` (page 9)
Returns a dictionary of the registered store types.

`+ registerStoreClass:forStoreType:` (page 9)
Registers a given `NSPersistentStore` subclass for a given store type string.

## Initializing a Coordinator

`- initWithManagedObjectModel:` (page 12)
Initializes the receiver with a managed object model.

`- managedObjectModel` (page 14)
Returns the receiver's managed object model.

## Configuring Persistent Stores

`- addPersistentStoreWithType:configuration:URL:options:error:` (page 10)
Adds a new persistent store of a specified type at a given location, and returns the new store.

`- setURL:forPersistentStore:` (page 18)
Sets the URL for a given persistent store.

`- removePersistentStore:error:` (page 17)
Removes a given persistent store.

`- migratePersistentStore:toURL:options:withType:error:` (page 15)
Moves a persistent store to a new location, changing the storage type if necessary.

`- persistentStores` (page 16)
Returns an array of persistent stores associated with the receiver.

`- persistentStoreForURL:` (page 16)
Returns the persistent store for the specified URL.

- `URLForPersistentStore:` (page 19)
  Returns the URL for a given persistent store.

## Locking

- `lock` (page 13)
  Attempts to acquire a lock.
- `tryLock` (page 19)
  Attempts to acquire a lock.
- `unlock` (page 19)
  Relinquishes a previously acquired lock.

## Working with Metadata

- `metadataForPersistentStore:` (page 14)
  Returns a dictionary that contains the metadata currently stored or to-be-stored in a given persistent store.
- `setMetadata:forPersistentStore:` (page 17)
  Sets the metadata stored in the persistent store during the next save operation executed on it to *`metadata`*.
+ `setMetadata:forPersistentStoreOfType:URL:error:` (page 10)
  Sets the metadata for a given store.
+ `metadataForPersistentStoreOfType:URL:error:` (page 8)
  Returns a dictionary containing the metadata stored in the persistent store at a given URL.
+ `metadataForPersistentStoreWithURL:error:` (page 29) Deprecated in Mac OS X v10.5
  Returns a dictionary that contains the metadata stored in the persistent store at the specified location. (Deprecated. Use `metadataForPersistentStoreOfType:URL:error:` (page 8) instead.)

## Working with Spotlight External Records

+ `elementsDerivedFromExternalRecordURL:` (page 8)
  Returns a dictionary containing the parsed elements derived from the Spotlight external record file specified by the given URL.
- `importStoreWithIdentifier:fromExternalRecordsDirectory:toURL:options:withType:error:` (page 11)
  Creates and populates a store with the external records found at a given URL.

## Discovering Object IDs

- `managedObjectIDForURIRepresentation:` (page 13)
  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.

# Class Methods

## elementsDerivedFromExternalRecordURL:

Returns a dictionary containing the parsed elements derived from the Spotlight external record file specified by the given URL.

```
+ (NSDictionary *)elementsDerivedFromExternalRecordURL:(NSURL *)fileURL
```

**Parameters**

*fileURL*

A file URL specifying the location of a Spotlight external record file.

**Return Value**

A dictionary containing the parsed elements derived from the Spotlight support file specified by *fileURL*.

**Discussion**

Dictionary keys and the corresponding values are described in "Spotlight External Record Elements" (page 25).

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

NSPersistentStoreCoordinator.h

## metadataForPersistentStoreOfType:URL:error:

Returns a dictionary containing the metadata stored in the persistent store at a given URL.

```
+ (NSDictionary *)metadataForPersistentStoreOfType:(NSString *)storeType URL:(NSURL
    *)url error:(NSError **)error
```

**Parameters**

*storeType*

The type of the store at *url*. If this value is nil, Core Data determines which store class should be used to get or set the store file's metadata by inspecting the file contents.

*url*

The location of a persistent store.

*error*

If no store is found at *url* or if there is a problem accessing its contents, upon return contains an NSError object that describes the problem.

**Return Value**

A dictionary containing the metadata stored in the persistent store at *url*, or nil if the store cannot be opened or if there is a problem accessing its contents.

The keys guaranteed to be in this dictionary are NSStoreTypeKey (page 21) and NSStoreUUIDKey (page 21).

**Discussion**
You can use this method to retrieve the metadata from a store without the overhead of creating a Core Data stack.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
`+ setMetadata:forPersistentStoreOfType:URL:error:` (page 10)
`- metadataForPersistentStore:` (page 14)
`- setMetadata:forPersistentStore:` (page 17)

**Declared In**
`NSPersistentStoreCoordinator.h`

## registeredStoreTypes

Returns a dictionary of the registered store types.

```
+ (NSDictionary *)registeredStoreTypes
```

**Return Value**
A dictionary of the registered store types—the keys are the store type strings, and the values are the `NSPersistentStore` subclasses.

**Availability**
Available in Mac OS X v10.5 and later.

**Declared In**
`NSPersistentStoreCoordinator.h`

## registerStoreClass:forStoreType:

Registers a given `NSPersistentStore` subclass for a given store type string.

```
+ (void)registerStoreClass:(Class)storeClass forStoreType:(NSString *)storeType
```

**Parameters**
*storeClass*
The `NSPersistentStore` subclass to use for the store of type *storeType*.
*storeType*
A unique string that identifies a store type.

**Discussion**
You must invoke this method before a custom subclass of `NSPersistentStore` can be loaded into a persistent store coordinator.

You can pass `nil` for *storeClass* to unregister the store type.

**Availability**
Available in Mac OS X v10.5 and later.

**Related Sample Code**
Core Data HTML Store

**Declared In**
`NSPersistentStoreCoordinator.h`

### setMetadata:forPersistentStoreOfType:URL:error:

Sets the metadata for a given store.

```
+ (BOOL)setMetadata:(NSDictionary *)metadata forPersistentStoreOfType:(NSString
    *)storeType URL:(NSURL *)url error:(NSError **)error
```

**Parameters**

*metadata*
A dictionary containing metadata for the store.

*storeType*
The type of the store at *url*. If this value is `nil`, Core Data will determine which store class should be used to get or set the store file's metadata by inspecting the file contents.

*url*
The location of a persistent store.

*error*
If no store is found at *url* or if there is a problem setting its metadata, upon return contains an `NSError` object that describes the problem.

**Return Value**
`YES` if the metadata was set correctly, otherwise `NO`.

**Discussion**
You can use this method to set the metadata for a store without the overhead of creating a Core Data stack.

**Availability**
Available in Mac OS X v10.5 and later.

**See Also**
`+ metadataForPersistentStoreOfType:URL:error:` (page 8)
`- metadataForPersistentStore:` (page 14)
`- setMetadata:forPersistentStore:` (page 17)

**Declared In**
`NSPersistentStoreCoordinator.h`

## 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.

```
- (NSPersistentStore *)addPersistentStoreWithType:(NSString *)storeType
    configuration:(NSString *)configuration URL:(NSURL *)storeURL
    options:(NSDictionary *)options error:(NSError **)error
```

**Parameters**

*storeType*

A string constant (such as `NSSQLiteStoreType`) that specifies the store type—see "Store Types" (page 20) for possible values.

*configuration*

The name of a configuration in the receiver's managed object model that will be used by the new store. The configuration can be `nil`, in which case no other configurations are allowed.

*storeURL*

The file location of the persistent store.

*options*

A dictionary containing key-value pairs that specify whether the store should be read-only, and whether (for an XML store) the XML file should be validated against the DTD before it is read. For key definitions, see "`Store Options`" (page 22) and "`Migration Options`" (page 23). This value may be `nil`.

*error*

If a new store cannot be created, upon return contains an instance of `NSError` that describes the problem

**Return Value**

The newly-created store or, if an error occurs, `nil`.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- `migratePersistentStore:toURL:options:withType:error:` (page 15)
- `importStoreWithIdentifier:fromExternalRecordsDirectory:toURL:options:withType:error:` (page 11)
- `removePersistentStore:error:` (page 17)

**Related Sample Code**

Core Data HTML Store

CoreRecipes

StickiesWithCoreData

**Declared In**

`NSPersistentStoreCoordinator.h`

## importStoreWithIdentifier:fromExternalRecordsDirectory:toURL:options:withType:error:

Creates and populates a store with the external records found at a given URL.

```
- (NSPersistentStore *)importStoreWithIdentifier:(NSString *)storeIdentifier
    fromExternalRecordsDirectory:(NSURL *)externalRecordsURL
    toURL:(NSURL *)destinationURL
    options:(NSDictionary *)options
    withType:(NSString *)storeType
    error:(NSError **)error
```

**Parameters**

*storeIdentifier*

The identifier for a store.

If this value is `nil` then the method imports the records for the first store found.

*externalRecordsURL*

The location of the directory containing external records.

*destinationURL*

An URL object that specifies the location for the new store.

There should be no existing store at this location, as the store will be created from scratch (appending to an existing store is not allowed).

*options*

A dictionary containing key-value pairs that specify whether the store should be read-only, and whether (for an XML store) the XML file should be validated against the DTD before it is read. For key definitions, see "Store Options" (page 22).

*storeType*

A string constant (such as `NSSQLiteStoreType`) that specifies the type of the new store—see "Store Types" (page 20).

*error*

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

**Return Value**

An object representing the newly-created store.

**Availability**

Available in Mac OS X v10.6 and later.

**See Also**

- `addPersistentStoreWithType:configuration:URL:options:error:` (page 10)
- `migratePersistentStore:toURL:options:withType:error:` (page 15)
- `removePersistentStore:error:` (page 17)

**Declared In**

`NSPersistentStoreCoordinator.h`

## initWithManagedObjectModel:

Initializes the receiver with a managed object model.

```
- (id)initWithManagedObjectModel:(NSManagedObjectModel *)model
```

**Parameters**

*model*

A managed object model.

**Return Value**
The receiver, initialized with *model*.

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

**Related Sample Code**
AbstractTree
Core Data HTML Store
CoreRecipes
Image Kit with Core Data
StickiesWithCoreData

**Declared In**
`NSPersistentStoreCoordinator.h`

## lock

Attempts to acquire a lock.

```
- (void)lock
```

**Discussion**
This method blocks a thread's execution until the lock can be acquired. An application protects a critical section of code by requiring a thread to acquire a lock before executing the code. Once the critical section is past, the thread relinquishes the lock by invoking unlock.

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

**See Also**
- `tryLock` (page 19)
- `unlock` (page 19)

**Declared In**
`NSPersistentStoreCoordinator.h`

## 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.

```
- (NSManagedObjectID *)managedObjectIDForURIRepresentation:(NSURL *)URL
```

**Parameters**
*URL*
An URL object containing a URI that specify a managed object.

**Return Value**
An object ID for the object specified by *URL*.

**Discussion**
The URI representation contains a UUID of the store the ID is coming from, and the coordinator can match it against the stores added to it.

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

**See Also**
`URIRepresentation` (`NSManagedObjectID`)
`objectWithID:` (`NSManagedObjectContext`)

**Related Sample Code**
CoreRecipes

**Declared In**
`NSPersistentStoreCoordinator.h`

## managedObjectModel

Returns the receiver's managed object model.

```
- (NSManagedObjectModel *)managedObjectModel
```

**Return Value**
The receiver's managed object model.

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

**Related Sample Code**
Core Data HTML Store
CoreRecipes
CustomAtomicStoreSubclass

**Declared In**
`NSPersistentStoreCoordinator.h`

## metadataForPersistentStore:

Returns a dictionary that contains the metadata currently stored or to-be-stored in a given persistent store.

```
- (NSDictionary *)metadataForPersistentStore:(NSPersistentStore *)store
```

**Parameters**
*store*
A persistent store.

**Return Value**
A dictionary that contains the metadata currently stored or to-be-stored in *store*.

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

**See Also**
- `setMetadata:forPersistentStore:` (page 17)
+ `metadataForPersistentStoreOfType:URL:error:` (page 8)
+ `setMetadata:forPersistentStoreOfType:URL:error:` (page 10)

**Related Sample Code**
CoreRecipes
Departments and Employees

**Declared In**
`NSPersistentStoreCoordinator.h`

## migratePersistentStore:toURL:options:withType:error:

Moves a persistent store to a new location, changing the storage type if necessary.

```
- (NSPersistentStore *)migratePersistentStore:(NSPersistentStore *)store toURL:(NSURL
    *)URL options:(NSDictionary *)options withType:(NSString *)storeType
   error:(NSError **)error
```

**Parameters**

*store*
A persistent store.

*URL*
An URL object that specifies the location for the new store.

*options*
A dictionary containing key-value pairs that specify whether the store should be read-only, and whether (for an XML store) the XML file should be validated against the DTD before it is read. For key definitions, see "Store Options" (page 22).

*storeType*
A string constant (such as `NSSQLiteStoreType`) that specifies the type of the new store—see "Store Types" (page 20).

*error*
If an error occurs, upon return contains an instance of `NSError` that describes the problem.

**Return Value**
If the migration is successful, the new store, otherwise `nil`.

**Discussion**
This method is typically used for "Save As" operations. Performance may vary depending on the type of old and new store. For more details of the action of this method, see Persistent Store Features in *Core Data Programming Guide*.

**Important:** After invocation of this method, the specified store is removed from the coordinator thus `store` is no longer a useful reference.

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

**See Also**
- `addPersistentStoreWithType:configuration:URL:options:error:` (page 10)
- `removePersistentStore:error:` (page 17)

**Related Sample Code**
File Wrappers with Core Data Documents

**Declared In**
`NSPersistentStoreCoordinator.h`

## persistentStoreForURL:

Returns the persistent store for the specified URL.

```
- (NSPersistentStore *)persistentStoreForURL:(NSURL *)URL
```

**Parameters**

*URL*

An URL object that specifies the location of a persistent store.

**Return Value**
The persistent store at the location specified by *URL*.

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

**See Also**
- `persistentStores` (page 16)
- `URLForPersistentStore:` (page 19)

**Related Sample Code**
CoreRecipes
Departments and Employees
File Wrappers with Core Data Documents

**Declared In**
`NSPersistentStoreCoordinator.h`

## persistentStores

Returns an array of persistent stores associated with the receiver.

```
- (NSArray *)persistentStores
```

**Return Value**
An array persistent stores associated with the receiver.

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

**See Also**
- `persistentStoreForURL:` (page 16)

- URLForPersistentStore: (page 19)

**Related Sample Code**
CoreRecipes

**Declared In**
NSPersistentStoreCoordinator.h

## removePersistentStore:error:

Removes a given persistent store.

```
- (BOOL)removePersistentStore:(NSPersistentStore *)store error:(NSError **)error
```

**Parameters**

*store*
A persistent store.

*error*
If an error occurs, upon return contains an instance of NSError that describes the problem.

**Return Value**
YES if the store is removed, otherwise NO.

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

**See Also**
- addPersistentStoreWithType:configuration:URL:options:error: (page 10)
- migratePersistentStore:toURL:options:withType:error: (page 15)

**Related Sample Code**
CoreRecipes
File Wrappers with Core Data Documents

**Declared In**
NSPersistentStoreCoordinator.h

## setMetadata:forPersistentStore:

Sets the metadata stored in the persistent store during the next save operation executed on it to *metadata*.

```
- (void)setMetadata:(NSDictionary *)metadata forPersistentStore:(NSPersistentStore
    *)store
```

**Parameters**

*metadata*
A dictionary containing metadata for the store.

*store*
A persistent store.

**Discussion**
The store type and UUID (`NSStoreTypeKey` and `NSStoreUUIDKey`) are always added automatically, however `NSStoreUUIDKey` is only added if it is not set manually as part of the dictionary argument.

**Important:** Setting the metadata for a store does not change the information on disk until the store is actually saved.

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

**See Also**
`- metadataForPersistentStore:` (page 14)
`+ setMetadata:forPersistentStoreOfType:URL:error:` (page 10)
`+ metadataForPersistentStoreOfType:URL:error:` (page 8)

**Related Sample Code**
CoreRecipes
Departments and Employees

**Declared In**
`NSPersistentStoreCoordinator.h`

## setURL:forPersistentStore:

Sets the URL for a given persistent store.

```
- (BOOL)setURL:(NSURL *)url forPersistentStore:(NSPersistentStore *)store
```

**Parameters**

*url*
The new location for *store*.

*store*
A persistent store associated with the receiver.

**Return Value**
`YES` if the store was relocated, otherwise `NO`.

**Discussion**
For atomic stores, this method alters the location to which the next save operation will write the file; for non-atomic stores, invoking this method will release the existing connection and create a new one at the specified URL. (For non-atomic stores, a store must already exist at the destination URL; a new store will not be created.)

**Availability**
Available in Mac OS X v10.5 and later.

**Related Sample Code**
File Wrappers with Core Data Documents

**Declared In**
`NSPersistentStoreCoordinator.h`

## tryLock

Attempts to acquire a lock.

```
- (BOOL)tryLock
```

**Return Value**
`YES` if successful, otherwise `NO`.

**Discussion**
Returns immediately—contrast lock (page 13) which blocks.

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

**See Also**
- `lock` (page 13)
- `unlock` (page 19)

**Declared In**
`NSPersistentStoreCoordinator.h`

## unlock

Relinquishes a previously acquired lock.

```
- (void)unlock
```

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

**See Also**
- `lock` (page 13)
- `tryLock` (page 19)

**Declared In**
`NSPersistentStoreCoordinator.h`

## URLForPersistentStore:

Returns the URL for a given persistent store.

```
- (NSURL *)URLForPersistentStore:(NSPersistentStore *)store
```

**Parameters**
*store*
A persistent store.

**Return Value**
The URL for *store*.

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

**See Also**
- `persistentStoreForURL:` (page 16)
- `persistentStores` (page 16)

**Related Sample Code**
CoreRecipes

**Declared In**
`NSPersistentStoreCoordinator.h`

# Constants

## Store Types

Types of persistent store.

```
NSString * const NSSQLiteStoreType;
NSString * const NSXMLStoreType;
NSString * const NSBinaryStoreType;
NSString * const NSInMemoryStoreType;
```

**Constants**

`NSSQLiteStoreType`
The SQLite database store type.
Available in Mac OS X v10.4 and later.
Declared in `NSPersistentStoreCoordinator.h`.

`NSXMLStoreType`
The XML store type.
Available in Mac OS X v10.4 and later.
Declared in `NSPersistentStoreCoordinator.h`.

`NSBinaryStoreType`
The binary store type.
Available in Mac OS X v10.4 and later.
Declared in `NSPersistentStoreCoordinator.h`.

`NSInMemoryStoreType`
The in-memory store type.
Available in Mac OS X v10.4 and later.
Declared in `NSPersistentStoreCoordinator.h`.

## Store Metadata

Keys used in a store's metadata dictionary.

```
NSString * const NSStoreTypeKey;
NSString * const NSStoreUUIDKey;
```

**Constants**

`NSStoreTypeKey`

The key in the metadata dictionary to identify the store type.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSStoreUUIDKey`

The key in the metadata dictionary to identify the store UUID.

The store UUID is useful to identify stores through URI representations, but it is *not* guaranteed to be unique. The UUID generated for new stores is unique—users can freely copy files and thus the UUID stored inside—so if you track or reference stores explicitly you need to be aware of duplicate UUIDs and potentially override the UUID when a new store is added to the list of known stores in your application.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

**Declared In**

`NSPersistentStoreCoordinator.h`

## Stores Change Notification User Info Keys

An `NSPersistentStoreCoordinatorStoresDidChangeNotification` (page 26) notification is posted whenever persistent stores are added to or removed from a persistent store coordinator, or when store UUIDs change. The *userInfo* dictionary contains information about the stores that were added or removed using these keys.

```
NSString * const NSAddedPersistentStoresKey;
NSString * const NSRemovedPersistentStoresKey;
NSString * const NSUUIDChangedPersistentStoresKey;
```

**Constants**

`NSAddedPersistentStoresKey`

Key for the array of stores that were added.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSRemovedPersistentStoresKey`

Key for the array of stores that were removed.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSUUIDChangedPersistentStoresKey`

Key for the array of stores whose UUIDs changed.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

**Declared In**

`NSPersistentStoreCoordinator.h`

## Store Options

Keys for the options dictionary used in `addPersistentStoreWithType:configuration:URL:options:error:` (page 10), `migratePersistentStore:toURL:options:withType:error:` (page 15), and `importStoreWithIdentifier:fromExternalRecordsDirectory:toURL:options:withType:error:` (page 11).

```
NSString * const NSReadOnlyPersistentStoreOption;
NSString * const NSValidateXMLStoreOption;
NSString * const NSPersistentStoreTimeoutOption;
NSString * const NSSQLitePragmasOption;
NSString * const NSSQLiteAnalyzeOption;
NSString * const NSSQLiteManualVacuumOption;
NSString * const NSExternalRecordsDirectoryOption;
NSString * const NSExternalRecordExtensionOption;
NSString * const NSExternalRecordsFileFormatOption;
```

### Constants

`NSReadOnlyPersistentStoreOption`

A flag that indicates whether a store is treated as read-only or not.

The default value is `NO`.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSValidateXMLStoreOption`

A flag that indicates whether an XML file should be validated with the DTD while opening.

The default value is `NO`.

Available in Mac OS X v10.4 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSPersistentStoreTimeoutOption`

Options key that specifies the connection timeout for Core Data stores.

The corresponding value is an `NSNumber` object that represents the duration in seconds that Core Data will wait while attempting to create a connection to a persistent store. If a connection is cannot be made within that timeframe, the operation is aborted and an error is returned.

Available in Mac OS X v10.5 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSSQLitePragmasOption`

Options key for a dictionary of SQLite pragma settings with pragma values indexed by pragma names as keys.

All pragma values must be specified as `NSString` objects. The `fullfsync` and `synchronous` pragmas control the tradeoff between write performance (write to disk speed & cache utilization) and durability (data loss/corruption sensitivity to power interruption). For more information on pragma settings, see http://sqlite.org/pragma.html.

Available in Mac OS X v10.5 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSSQLiteAnalyzeOption`
: Option key to run an analysis of the store data to optimize indices based on statistical information when the store is added to the coordinator.

  This invokes SQLite's `ANALYZE` command. It is ignored by stores other than the SQLite store.

  Available in Mac OS X v10.6 and later.

  Declared in `NSPersistentStoreCoordinator.h`.

`NSSQLiteManualVacuumOption`
: Option key to rebuild the store file, forcing a database wide defragmentation when the store is added to the coordinator.

  This invokes SQLite's `VACUUM` command. It is ignored by stores other than the SQLite store.

  Available in Mac OS X v10.6 and later.

  Declared in `NSPersistentStoreCoordinator.h`.

`NSExternalRecordsDirectoryOption`
: Option indicating the directory where Spotlight external record files should be written to.

  Available in Mac OS X v10.6 and later.

  Declared in `NSPersistentStoreCoordinator.h`.

`NSExternalRecordExtensionOption`
: Option indicating the file extension to use for Spotlight external record files.

  Available in Mac OS X v10.6 and later.

  Declared in `NSPersistentStoreCoordinator.h`.

`NSExternalRecordsFileFormatOption`
: Option to specify the file format of a Spotlight external records.

  For possible values, see "Spotlight External Record File Format Options" (page 24).

  Available in Mac OS X v10.6 and later.

  Declared in `NSPersistentStoreCoordinator.h`.

## Migration Options

Migration options, specified in the dictionary of options when adding a persistent store using `addPersistentStoreWithType:configuration:URL:options:error:` (page 10).

```
NSString * const NSIgnorePersistentStoreVersioningOption;
NSString * const NSMigratePersistentStoresAutomaticallyOption;
NSString * const NSInferMappingModelAutomaticallyOption;
```

### Constants

`NSIgnorePersistentStoreVersioningOption`
: Key to ignore the built-in versioning provided by Core Data.

  The corresponding value is an `NSNumber` object. If the `boolValue` of the number is `YES`, Core Data will not compare the version hashes between the managed object model in the coordinator and the metadata for the loaded store. (It will, however, continue to update the version hash information in the metadata.) This key and corresponding value of `YES` is specified by default for all applications linked on or before Mac OS X 10.4.

  Available in Mac OS X v10.5 and later.

  Declared in `NSPersistentStoreCoordinator.h`.

`NSMigratePersistentStoresAutomaticallyOption`
Key to automatically attempt to migrate versioned stores.

The corresponding value is an `NSNumber` object. If the `boolValue` of the number is `YES` and if the version hash information for the added store is determined to be incompatible with the model for the coordinator, Core Data will attempt to locate the source and mapping models in the application bundles, and perform a migration.

Available in Mac OS X v10.5 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSInferMappingModelAutomaticallyOption`
Key to attempt to create the mapping model automatically.

The corresponding value is an `NSNumber` object. If the `boolValue` of the number is `YES` and the value of the `NSMigratePersistentStoresAutomaticallyOption` is `YES`, the coordinator will attempt to infer a mapping model if none can be found.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

## Spotlight External Record File Format Options

Constants that specify the format for Spotlight external records. These constants are the possible values for the key `NSExternalRecordsFileFormatOption` (page 23).

```
NSString * const NSXMLExternalRecordType;
NSString * const NSBinaryExternalRecordType;
```

**Constants**

`NSXMLExternalRecordType`
Specifies an XML file format.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSBinaryExternalRecordType`
Specifies a binary file format

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

## Versioning Support

Keys in store metadata to support versioning.

```
NSString * const NSStoreModelVersionHashesKey;
NSString * const NSStoreModelVersionIdentifiersKey;
NSString * const NSPersistentStoreOSCompatibility;
```

**Constants**

`NSStoreModelVersionHashesKey`

Key to represent the version hash information for the model used to create the store.

This key is used in the metadata for a persistent store.

Available in Mac OS X v10.5 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSStoreModelVersionIdentifiersKey`

Key to represent the version identifiers for the model used to create the store.

If you add your own annotations to a model's version identifier (see `versionIdentifiers`), they are stored in the persistent store's metadata. You can use this key to retrieve the identifiers from the metadata dictionaries available from `NSPersistentStore` (`metadata`) and `NSPersistentStoreCoordinator` (`metadataForPersistentStore:` (page 14) and related methods). The corresponding value is a Foundation collection (an `NSArray` or `NSSet` object).

Available in Mac OS X v10.5 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSPersistentStoreOSCompatibility`

Key to represent the earliest version of Mac OS X the persistent store supports.

The corresponding value is an `NSNumber` object that takes the form of the constants defined by the Mac OS X availability macros (defined in `/usr/include/AvailabilityMacros.h`), for example `1040` represents Mac OS X version 10.4.0.

Backward compatibility may preclude some features.

Available in Mac OS X v10.5 and later.

Declared in `NSPersistentStoreCoordinator.h`.

## Spotlight External Record Elements

Keys for the dictionary with the parsed elements derived from Spotlight external record file.

```
NSString * const NSEntityNameInPathKey;
NSString * const NSStoreUUIDInPathKey;
NSString * const NSStorePathKey;
NSString * const NSModelPathKey;
NSString * const NSObjectURIKey;
```

**Constants**

`NSEntityNameInPathKey`

Dictionary key for the entity name extracted from an external record file.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSStoreUUIDInPathKey`

Dictionary key for the store UUID extracted from an external record file.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSStorePathKey`

Dictionary key for the store path (an instance of `NSURL`) extracted from an external record file.

This is resolved to the store-file path contained in the an external record file directory.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSModelPathKey`

Dictionary key for the managed object model path (an instance of `NSURL`) extracted from an external record file.

This is resolved to the `model.mom` path contained in the external record file directory.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

`NSObjectURIKey`

Dictionary key for the object URI extracted from an external record file.

Available in Mac OS X v10.6 and later.

Declared in `NSPersistentStoreCoordinator.h`.

# Notifications

### NSPersistentStoreCoordinatorStoresDidChangeNotification

Posted whenever persistent stores are added to or removed from a persistent store coordinator, or when store UUIDs change.

The notification's object is the persistent store coordinator that was affected. The notification's *userInfo* dictionary contains information about the stores that were added or removed, specified using the following keys:

| | |
|---|---|
| `NSAddedPersistentStoresKey` (page 21) | Key for the array of stores that were added. |
| `NSRemovedPersistentStoresKey` (page 21) | Key for the array of stores that were removed. |
| `NSUUIDChangedPersistentStoresKey` (page 21) | Key for the array of stores whose UUIDs changed. |

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

**Declared In**
`NSPersistentStoreCoordinator.h`

### NSPersistentStoreCoordinatorWillRemoveStoreNotification

Posted whenever a persistent store is removed from a persistent store coordinator.

The notification is sent during the invocation of `NSPersistentStore`'s `willRemoveFromPersistentStoreCoordinator` method during store deallocation or removal. The notification's object is the persistent store coordinator will be removed.

**Availability**
Available in Mac OS X v10.6 and later.

**Declared In**
`NSPersistentStoreCoordinator.h`

# Deprecated NSPersistentStoreCoordinator Methods

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

## Deprecated in Mac OS X v10.5

### metadataForPersistentStoreWithURL:error:

Returns a dictionary that contains the metadata stored in the persistent store at the specified location. (Deprecated in Mac OS X v10.5. Use `metadataForPersistentStoreOfType:URL:error:` (page 8) instead.)

```
+ (NSDictionary *)metadataForPersistentStoreWithURL:(NSURL *)url error:(NSError
    **)error
```

**Parameters**

*url*

An URL object that specifies the location of a persistent store.

*error*

If no store is found at *url* or if there is a problem accessing its contents, upon return contains an instance of `NSError` that describes the problem.

**Return Value**

A dictionary containing the metadata for the persistent store at *url*. If no store is found, or there is a problem accessing its contents, returns `nil`.

The keys guaranteed to be in this dictionary are `NSStoreTypeKey` and `NSStoreUUIDKey`.

**Discussion**

This method allows you to access the metadata in a persistent store without initializing a Core Data stack.

**Availability**

Available in Mac OS X v10.4 and later.

Deprecated in Mac OS X v10.5.

**See Also**

`- metadataForPersistentStore:` (page 14)

`- setMetadata:forPersistentStore:` (page 17)

`+ metadataForPersistentStoreOfType:URL:error:` (page 8)

`+ setMetadata:forPersistentStoreOfType:URL:error:` (page 10)

**Related Sample Code**

CoreRecipes

**Declared In**
`NSPersistentStoreCoordinator.h`

# Document Revision History

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

| Date | Notes |
| --- | --- |
| 2009-05-01 | Corrected typographical errors. |
| 2009-02-12 | Updated for Mac OS X v10.6. |
| 2009-02-03 | Updated for iOS 3.0. |
| 2008-03-11 | Corrected minor typographical errors. |
| 2007-04-09 | Updated for Mac OS X v10.5. |
| 2006-05-23 | First publication of this content as a separate document. |

Document Revision History