
Sync Services Framework Reference

[Cocoa > Syncing](#)



2007-10-31



Apple Inc.
© 2004, 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

.Mac is a registered service mark of Apple Inc.

Apple, the Apple logo, Cocoa, iCal, iPod, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction **Introduction 7**

Part I **Classes 9**

Chapter 1 **ISyncChange Class Reference 11**

Overview 11
Tasks 12
Class Methods 12
Instance Methods 13
Constants 15

Chapter 2 **ISyncClient Class Reference 17**

Overview 17
Tasks 18
Instance Methods 20
Constants 31

Chapter 3 **ISyncFilter Class Reference 33**

Overview 33
Tasks 33
Class Methods 34

Chapter 4 **ISyncManager Class Reference 35**

Overview 35
Tasks 36
Class Methods 37
Instance Methods 37
Constants 42
Notifications 42

Chapter 5 **ISyncRecordReference Class Reference 43**

Overview 43

Chapter 6 **ISyncRecordSnapshot Class Reference 45**

Overview 45

Tasks 46
Instance Methods 46

Chapter 7 **ISyncSession Class Reference 51**

Overview 51
Tasks 53
Class Methods 55
Instance Methods 61
Constants 75

Chapter 8 **ISyncSessionDriver Class Reference 77**

Overview 77
Tasks 78
Class Methods 79
Instance Methods 80
Delegate Methods 84

Chapter 9 **NSPersistentStoreCoordinator Sync Services Additions Reference 89**

Overview 89
Tasks 89
Instance Methods 90

Part II **Managers 93**

Chapter 10 **Sync Services Constants Reference 95**

Overview 95
Constants 95

Part III **Protocols 101**

Chapter 11 **ISyncFiltering Protocol Reference 103**

Overview 103
Tasks 103
Instance Methods 104

Chapter 12 **ISyncSessionDriverDataSource Protocol Reference 107**

Overview 107
Tasks 108
Instance Methods 109

Constants 118

Chapter 13 **ISyncUIHelper Protocol Reference 119**

Overview 119

Tasks 119

Instance Methods 120

Chapter 14 **NSPersistentStoreCoordinatorSyncing Protocol Reference 123**

Overview 123

Tasks 123

Instance Methods 124

Document Revision History 133

Index 135

Introduction

Framework	/System/Library/Frameworks/SyncServices.framework
Header file directories	/System/Library/Frameworks/SyncServices.framework/Headers
Declared in	ISyncChange.h ISyncClient.h ISyncCoreData.h ISyncFilter.h ISyncManager.h ISyncRecordSnapshot.h ISyncSession.h ISyncSessionDriver.h SyncServicesErrors.h

The Sync Services framework provides all the classes and protocols you need to sync your application's data with other applications and devices on the same computer or—using a .Mac account—with applications and devices on other computers. Developers use this framework to manage their sync sessions and communicate with the sync engine to push and pull changes.

Classes

ISyncChange Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncChange.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide
Related sample code	People SeeMyFriends StickiesExample

Overview

An `ISyncChange` object encapsulates a set of changes to a single record such as adding, deleting, and modifying a record. You use `ISyncChange` objects to push your changes to the sync engine. Similarly, you pull `ISyncChange` objects from the sync engine when applying sync engine changes.

You use the [changeWithType:recordIdentifier:changes:](#) (page 12) method to create an `ISyncChange` object specifying the type of change (an add, delete, or modify), the identifier for the record that changed, and the new property values if any. Use the [pushChange:](#) (page 71) `ISyncSession` method to push the change to the sync engine. You can also use the [deleteRecordWithIdentifier:](#) (page 68) `ISyncSession` method to delete a record without needing to create an `ISyncChange` object.

Use the [changeEnumeratorForEntityNames:](#) (page 61) `ISyncSession` method to pull changes from the sync engine. Use the returned object enumerator to process each `ISyncChange` object. Use the [type](#) (page 14) method to each `ISyncChange` object to get the type of change (add, modify or delete), the [recordIdentifier](#) (page 14) method to get the record identifier, and the [changes](#) (page 13) method get descriptions of the change.

Tasks

Creating and Initializing Instances

- + [changeWithType:recordIdentifier:changes:](#) (page 12)
Creates an ISyncChange object.
- [initWithChangeType:recordIdentifier:changes:](#) (page 13)
Initializes an ISyncChange object.

Getting Attributes

- [type](#) (page 14)
Returns the type of change the receiver represents.
- [recordIdentifier](#) (page 14)
Returns the unique record identifier for the record that changed.
- [record](#) (page 14)
Returns a dictionary representation of the record that changed.
- [changes](#) (page 13)
Returns an array of changes made to the record.

Class Methods

changeWithType:recordIdentifier:changes:

Creates an ISyncChange object.

```
+ (id)changeWithType:(ISyncChangeType)type recordIdentifier:(NSString *)recordIdentifier changes:(NSArray *)changes
```

Discussion

Creates and returns an ISyncChange object of the type specified by *type*, for the record identified by *recordIdentifier*, and with the changes specified by *changes*. The *type* argument should be one of the [ISyncChangeTypeAdd](#) (page 15), [ISyncChangeTypeModify](#) (page 15) or [ISyncChangeTypeDelete](#) (page 15) constants. The *changes* array encapsulates multiple property changes to a single record—it is expected to contain dictionaries that use the keys described in “[Constants](#)” (page 15). Each dictionary in the array encapsulates the change to a single property and specifies the property name, action, and new value if applicable. Use this method to create changes for pushing to the sync engine. Use the [pushChange:](#) (page 71) ISyncSession method to push the change to the sync engine.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncChange.h

Instance Methods

changes

Returns an array of changes made to the record.

- (NSArray *)changes

Discussion

Returns an array of the changes made to the record returned by [record](#) (page 14) with the identifier returned by [recordIdentifier](#) (page 14). The changes array encapsulates multiple property changes to a single record. The returned array contains dictionaries with keys specifying the type of change to a property and its new value. See “[Constants](#)” (page 15) for a description of the keys used in these dictionaries. See [ISyncChangePropertyValueKey](#) (page 16) for a description of the value of relationship properties. Returns `nil` if the change type is `ISyncChangeTypeDelete`.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [type](#) (page 14)

Related Sample Code

StickiesExample

Declared In

ISyncChange.h

initWithChangeType:recordIdentifier:changes:

Initializes an ISyncChange object.

```
- (id)initWithChangeType:(ISyncChangeType)type recordIdentifier:(NSString *)recordIdentifier changes:(NSArray *)changes
```

Discussion

Initializes an ISyncChange object of the type specified by *type*, for the record identified by *recordIdentifier*, and with the changes specified by *changes*. The *type* argument should be one of the [ISyncChangeTypeAdd](#) (page 15), [ISyncChangeTypeModify](#) (page 15) or [ISyncChangeTypeDelete](#) (page 15) constants. The *changes* array encapsulates multiple property changes to a single record—it is expected to contain dictionaries that use the keys described in “[Constants](#)” (page 15). Each dictionary encapsulates the change to a single property and specifies the property name, action, and new value if applicable. Use this method to create changes for pushing to the sync engine. Use the [pushChange:](#) (page 71) `ISyncSession` method to push the change to the sync engine. This is the designated initializer for this class.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncChange.h

record

Returns a dictionary representation of the record that changed.

- (NSDictionary *)record

Discussion

The dictionary contains a key-value pair for each property unless the value of a property is unspecified. Only changes created by the sync engine have an associated record. Returns `nil` if the client created the receiver to push changes, or this is a delete change.

When pulling changes, this method returns a copy of the record as it appears in the truth database. Only those properties supported by the client are included in this record. Use the [changes](#) (page 13) method to get the changes made to this record, and use the [recordIdentifier](#) (page 14) method to get the record's unique identifier.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

People

StickiesExample

Declared In

ISyncChange.h

recordIdentifier

Returns the unique record identifier for the record that changed.

- (NSString *)recordIdentifier

Discussion

Returns the unique identifier for the record returned by [record](#) (page 14), or the record identifier you specified when creating the receiver. Use the [changes](#) (page 13) method to get the changes that were made to this record.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

People

SeeMyFriends

StickiesExample

Declared In

ISyncChange.h

type

Returns the type of change the receiver represents.

- (ISyncChangeType)type

Discussion

Returns whether or not this change was an add, delete or modify. See “[Constants](#)” (page 15) for a description of the possible return values.

Availability

Available in Mac OS X v10.4 and later.

See Also

– [changes](#) (page 13)

Declared In

ISyncChange.h

Constants

The following constants are used to determine the type of change. Use one of these values to set the *type* argument when creating an ISyncChange object using either the [changeWithType:recordIdentifier:changes:](#) (page 12) class method or the [initWithChangeType:recordIdentifier:changes:](#) (page 13) instance method. The [type](#) (page 14) method also returns one of these values.

Constant	Description
ISyncChangeTypeAdd	Indicates a record was added. Available in Mac OS X v10.4 and later. Declared in ISyncChange.h.
ISyncChangeTypeModify	Indicates a record was modified. Available in Mac OS X v10.4 and later. Declared in ISyncChange.h.
ISyncChangeTypeDelete	Indicates a record was deleted. Available in Mac OS X v10.4 and later. Declared in ISyncChange.h.

The following constants are used as keys for individual property changes encapsulated in an ISyncChange object. When pushing changes, use these keys to set key-value pairs for dictionaries you add to the *changes* argument passed to either the [changeWithType:recordIdentifier:changes:](#) (page 12) class method or the [initWithChangeType:recordIdentifier:changes:](#) (page 13) instance method. When pulling changes, you also use these keys to get the attributes of each change. Use the object enumerator returned by the [changes](#) (page 13) method to iterate through the changes array.

Constant	Description
ISyncChangePropertyActionKey	Specifies whether or not the property is being set or deleted. The value for this key should be either ISyncChangePropertySet or ISyncChangePropertyClear described below.

Constant	Description
<code>ISyncChangeProperty-NameKey</code>	Key for the name of the property.
<code>ISyncChangeProperty-ValueKey</code>	Key for the new value of the property. Not used if the action is <code>ISyncChangePropertyClear</code> . However, the absence of this key does not imply the property is being deleted. This key-value pair may be omitted if the value is unspecified. You can also set the value to <code>nil</code> . If the property is a relationship, then the value is an array of record identifiers belonging to the destination objects. If the relationship is to-one, this array contains a single record identifier.

The following constants are possible values for the `ISyncChangePropertyActionKey` (page 15) key used to describe the type of change to single property.

Constant	Description
<code>ISyncChangePropertySet</code>	Indicates the property was modified.
<code>ISyncChangePropertyClear</code>	Indicates the property was deleted.

ISyncClient Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncClient.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide
Related sample code	People SeeMyFriends StickiesExample

Overview

An `ISyncClient` object represents an application, tool, or device that syncs records—for example, Address Book, .Mac, or a mobile phone.

An `ISyncClient` object encapsulates information that assists the sync engine in identifying your client, determining its capabilities, and maintaining its state. For example, you use an `ISyncClient` object to get the list of entities that a client supports, find out when an entity was last synced, and setup filters. `ISyncClient` also provides some methods for controlling the sync mode.

You create an `ISyncClient` object by registering a unique client identifier with the shared `ISyncManager` object. Send either the `registerClientWithIdentifier:descriptionFilePath:` (page 38) or `clientWithIdentifier:` (page 38) message to the shared `ISyncManager` object. You obtain the shared instance by sending `sharedManager` (page 37) to `ISyncManager` class. You unregister a client, remove all information the sync engine knows about that client, using the `unregisterClient:` (page 41) `ISyncManager` method. See *Sync Services Programming Guide* for more information on registering and unregistering clients. You should never subclass or instantiate `ISyncClient` directly.

When you create a client using the `registerClientWithIdentifier:descriptionFilePath:` (page 38) `ISyncManager` method, you specify the client's capabilities using a client description file. Some of the `ISyncClient` methods are simply accessors that you can use to get or set the properties of this client description. For example, you use the client description to specify the entities and properties that a client supports, and you use the `supportedEntityNames` (page 30) method to get those supported entities. You can also use the `canPushChangesForEntityName:` (page 20) and `canPullChangesForEntityName:` (page 20) methods to find out which entities your client can push and pull. See *Sync Services Programming Guide* to learn more about the properties of a client description file.

Typically, the user requests that an application or device be resets (so that all the records on the client are replaced by the records in the truth database). The preference panel or configuration tool that receives this user request sends a [setShouldReplaceClientRecords:forEntityNames:](#) (page 27) message to the ISyncClient so that the next time the client syncs the truth is pulled. This is called a **pull the truth** sync mode and must be requested before the sync session enters the negotiation state.

Clients can optionally sync simultaneously. Use the [setShouldSynchronize:withClientsOfType:](#) (page 27) method to specify the type of client your client is interested in syncing simultaneously with. If you want to participate in a sync when your application isn't running, use the [setSyncAlertToolPath:](#) (page 29) method to specify that an alert tool be launched. Otherwise, use the [setSyncAlertHandler:selector:](#) (page 28) method to specify that a target and action be invoked when another client of the specified type syncs. If both a sync tool and sync target-action are registered, only the sync target-action is invoked.

If your application uses only a subset of the entities, attributes, and relationships defined in a schema, then you can restrict pulled records to that subset using custom filters. You set filters using the [setFilters:](#) (page 25) method. Each filter is expected to conform to the ISyncFiltering protocol and are used to reject or accept records from the sync engine before they are pulled. Use the [filters](#) (page 22) method to get the filters currently used by a client. See *Sync Services Programming Guide* for more information on using filters.

Tasks

Getting and Setting Attributes

- [clientIdentifier](#) (page 21)
Returns the client's identifier specified when registering the client.
- [clientType](#) (page 21)
Returns the receiver's client type.
- [displayName](#) (page 21)
Returns the receiver's display name specified in the client description file when registering the client or by sending [setDisplayDisplayName:](#) (page 24) to the receiver.
- [imagePath](#) (page 22)
Returns the absolute path to the image representation of the client.
- [setDisplayDisplayName:](#) (page 24)
Sets the display name for the receiver to *displayName*.
- [setImagePath:](#) (page 26)
Sets the receiver's absolute image path to *path*.
- [objectForKey:](#) (page 24)
Returns the object for *key* that was specified using the [setObject:forKey:](#) (page 26) method.
- [setObject:forKey:](#) (page 26)
Associates arbitrary information specified by a key-value pair to the receiver.

Specifying Supported Entities

- [canPushChangesForEntityName:](#) (page 20)
Returns YES if the client supports pushing changes to entity records specified by *entityName*, NO otherwise.
- [canPullChangesForEntityName:](#) (page 20)
Returns YES if the client supports pulling changes to entity records specified by *entityName*, NO otherwise.
- [supportedEntityNames](#) (page 30)
Returns an array of NSString objects containing the names of the entities the client supports.

Getting Sync Status

- [lastSyncDateForEntityName:](#) (page 23)
Returns the start date of the last time an entity, specified by *entityName*, was synced.
- [lastSyncStatusForEntityName:](#) (page 24)
Returns the status of the last time an entity, specified by *entityName* was synced.

Enabling Entities

- [enabledEntityNames](#) (page 22)
Returns an array of NSString objects containing the names of the entities that are enabled.
- [isEnabledForEntityName:](#) (page 23)
Returns YES if the entity specified by *entityName* is enabled, NO otherwise.
- [setEnabled:forEntityNames:](#) (page 25)
If *flag* is YES, enables the entities specified by *entityNames*, otherwise disables them.

Replacing Records

- [setShouldReplaceClientRecords:forEntityNames:](#) (page 27)
Sets whether or not a client should pull the truth—replace all its records for the specified entities on the next sync.
- [shouldReplaceClientRecordsForEntityName:](#) (page 30)
Returns YES if the client should replace all records for the entity specified by *entityName* during the next sync, NO otherwise.

Filtering

- [filters](#) (page 22)
Returns an array of filters that define a subset of the records the client syncs.
- [setFilters:](#) (page 25)
Sets the receiver's filters used to control the records pulled from the sync engine to *filters*, an array of objects conforming to the ISyncFiltering protocol.

Alerting Clients

- [shouldSynchronizeWithClientsOfType:](#) (page 30)
Returns YES if the client is registered to receive alerts when clients of *clientType* sync, NO otherwise.
- [setShouldSynchronize:withClientsOfType:](#) (page 27)
Adds the receiver as an observer of alerts when clients of the specified type sync.
- [setSyncAlertToolPath:](#) (page 29)
Specifies the absolute path to a tool that is launched when an observed client creates a session and begins syncing.
- [setSyncAlertHandler:selector:](#) (page 28)
Sets the target and action to be invoked when an observed client creates a session and begins syncing.
- [syncAlertToolPath](#) (page 31)
Returns the path to the tool that is launched when an observed client begins syncing.

Instance Methods

canPullChangesForEntityName:

Returns YES if the client supports pulling changes to entity records specified by *entityName*, NO otherwise.

```
- (BOOL)canPullChangesForEntityName:(NSString *)entityName
```

Discussion

Use this method to determine if a client is capable of pulling entity records. For example, an iPod or phone client might pull but never push changes to contacts and calendars. This property is set when registering the client.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [canPushChangesForEntityName:](#) (page 20) (ISyncManager)
- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38)

Declared In

ISyncClient.h

canPushChangesForEntityName:

Returns YES if the client supports pushing changes to entity records specified by *entityName*, NO otherwise.

```
- (BOOL)canPushChangesForEntityName:(NSString *)entityName
```

Discussion

Use this method to determine if a client is capable of pushing entity records. For example, an iPod or phone client might pull but never push changes to contacts and calendars. This property is set when registering the client.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [canPullChangesForEntityName:](#) (page 20) ([ISyncManager](#))
- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38)

Declared In

`ISyncClient.h`

clientIdentifier

Returns the client's identifier specified when registering the client.

```
- (NSString *)clientIdentifier
```

Discussion

You set the client identifier when registering the client using the [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) [ISyncManager](#) method. The client identifier is expected to be unique across all clients and is typically a DNS-style name.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`ISyncClient.h`

clientType

Returns the receiver's client type.

```
- (NSString *)clientType
```

Discussion

The returned string is expected to be one of the constants described in "[Constants](#)" (page 31). The client type is used to match clients that want to sync simultaneously. You specify the client type in the client description file when registering the client using the [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) [ISyncManager](#) method.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`ISyncClient.h`

displayName

Returns the receiver's display name specified in the client description file when registering the client or by sending [setDisplayDisplayName:](#) (page 24) to the receiver.

```
- (NSString *)displayName
```

Availability

Available in Mac OS X v10.4 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) (ISyncManager)

Declared In

ISyncClient.h

enabledEntityNames

Returns an array of NSString objects containing the names of the entities that are enabled.

- (NSArray*)enabledEntityNames

Discussion

The enabled entities may be a subset of the supported entities. Use [setEnabled:forEntityNames:](#) (page 25) to enable or disable an entity. You should pass the returned array as the *entityNames* argument to one of the [beginTransactionWithClient...](#) ISyncSession class methods when creating a session.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [isEnabledForEntityName:](#) (page 23)

Declared In

ISyncClient.h

filters

Returns an array of filters that define a subset of the records the client syncs.

- (NSArray*)filters

Discussion

Objects in the returned array are expected to conform to the ISyncFiltering protocol.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setFilters:](#) (page 25)

Declared In

ISyncClient.h

imagePath

Returns the absolute path to the image representation of the client.

- (NSString *)imagePath

Discussion

You can specify an image path in the client description file when registering a client or by sending [setImagePath:](#) (page 26) to the receiver.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) (ISyncManager)

Related Sample Code

StickiesExample

Declared In

ISyncClient.h

isEnabledForEntityName:

Returns YES if the entity specified by *entityName* is enabled, NO otherwise.

- (BOOL)isEnabledForEntityName:(NSString *)entityName

Discussion

If this method returns NO, the sync engine does not allow the client to sync records of type *entityName*.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [enabledEntityNames](#) (page 22)

- [setEnabled:forEntityNames:](#) (page 25)

Declared In

ISyncClient.h

lastSyncDateForEntityName:

Returns the start date of the last time an entity, specified by *entityName*, was synced.

- (NSDate *)lastSyncDateForEntityName:(NSString *)entityName

Discussion

Returns a start date of the last sync even if the last sync failed. Returns the start date of the previous sync if the client is currently syncing the entity. Returns *nil* if the client never synced the specified entity or the entity is not supported.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [lastSyncStatusForEntityName:](#) (page 24)

Declared In

ISyncClient.h

lastSyncStatusForEntityName:

Returns the status of the last time an entity, specified by *entityName* was synced.

```
- (ISyncStatus)lastSyncStatusForEntityName:(NSString *)entityName
```

Discussion

For example, the last sync may have succeeded, may have failed, may be in progress, or may have been canceled (see “[Constants](#)” (page 31) for other possible return values). Returns [ISyncStatusNever](#) (page 32) if the client never synced the specified entity, or the entity is not supported.

The sync engine maintains the last sync information for as long as the client supports *entityName*. When a client stops supporting *entityName*, the last sync information for that entity is removed. If the client starts supporting *entityName* again, this method behaves as if the client never synced the entity.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [lastSyncDateForEntityName:](#) (page 23)

Declared In

ISyncClient.h

objectForKey:

Returns the object for *key* that was specified using the [setObject:forKey:](#) (page 26) method.

```
- (id)objectForKey:(NSString *)key
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncClient.h

setDisplayName:

Sets the display name for the receiver to *displayName*.

```
- (void)setDisplayName:(NSString *)displayName
```

Discussion

The display name may be used by GUI applications to graphically identify the client to users. You can also specify a display name when registering the client using the client description file.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) (ISyncManager)
- [displayName](#) (page 21)

Declared In

ISyncClient.h

setEnabled:forEntityNames:

If *flag* is YES, enables the entities specified by *entityNames*, otherwise disables them.

```
- (void)setEnabled:(BOOL)flag forEntityNames:(NSArray *)entityNames
```

Discussion

The *entityNames* array of NSString objects is expected to contain names of supported entities, otherwise an exception is raised.

The first time a client syncs, a panel appears asking the user if it's OK to sync entities belonging to a data class (a panel may appear for each data class). If the user declines then the entities are disabled, otherwise they are enabled. If you want to allow the user to enable entities, invoke this method by passing YES as the *flag* argument and all the entity names in the data class as the *entityNames* argument. Then the next time the client syncs, a panel appears again asking the user if it's OK to sync the data class.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [enabledEntityNames](#) (page 22)
- [isEnabledForEntityName:](#) (page 23)

Declared In

ISyncClient.h

setFilters:

Sets the receiver's filters used to control the records pulled from the sync engine to *filters*, an array of objects conforming to the ISyncFiltering protocol.

```
- (void)setFilters:(NSArray *)filters
```

Discussion

You use filters to define a subset of the records that this client syncs.

When pulling changes, the sync engine passes each record to each filter before giving changes to that record to the client. If any one of the filters rejects the record, it is not given to the client. See *ISyncFilter Class Reference* for some default filters.

This method recomputes the records that need to be sent to the client during the next sync operation which can be expensive. Consequently, do not invoke this method frequently.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [filters](#) (page 22)

Related Sample Code

People

Declared In

ISyncClient.h

setImagePath:

Sets the receiver's absolute image path to *path*.

```
- (void)setImagePath:(NSString *)path
```

Discussion

The image may be used by GUI applications to represent the client. You can also specify an image path when registering the client using the client description file.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [imagePath](#) (page 22) (ISyncManager)

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38)

Related Sample Code

StickiesExample

Declared In

ISyncClient.h

setObject:forKey:

Associates arbitrary information specified by a key-value pair to the receiver.

```
- (void)setObject:(id < NSCoding >)value forKey:(NSString *)key
```

Discussion

This method retains *value* and copy *key*. Pass *nil* for *value* to release a previously retained value. Use [objectForKey:](#) (page 24) to retrieve the value for a given key. The *value* is released when the client is unregistered.

This method is provided as a convenience for developers who have additional data they want to store with an object that is not defined in the schema. For example, use this method to store client-specific configuration information if multiple clients are associated with the same user defaults domain or if you want to store a sync anchor.

A sync anchor is an identifier exchanged between a client and a device, or between two clients running on different computers. Typically, the client that initiates a sync is passed a sync anchor to the device or another client at the end of a successful sync. The next time the client syncs, the recipient of the sync anchor passes the anchor back to the original client to verify that it is in a known state.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncClient.h

setShouldReplaceClientRecords:forEntityNames:

Sets whether or not a client should pull the truth—replace all its records for the specified entities on the next sync.

```
- (void)setShouldReplaceClientRecords:(BOOL)flag forEntityNames:(NSArray
    *)entityNames
```

Discussion

If *flag* is YES, the client should replace all its local records with the records pulled from the sync engine.

After invoking this method, sending `shouldReplaceClientRecordsForEntityName:` (page 30) to any new sessions created for this client returns YES, and sending `shouldPushChangesForEntityName:` (page 74) or `shouldPushAllRecordsForEntityName:` (page 73) returns NO.

This request takes effect on the next session created after invoking this method and remains in effect until the client successfully passes through the pull phase of that session. The sync engine needs to know whether a client is going to pull the truth before entering the negotiation phase. This is necessary to detect conflicting push the truth and pull the truth requests.

A client should not remove its local records until after the records are successfully pulled from the sync engine. The local records can be safely removed after `shouldReplaceClientRecordsForEntityName:` (page 30) returns YES.

This method is typically used by a configuration tool that allows the user to revert to the state of the truth.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

People

Declared In

ISyncClient.h

setShouldSynchronize:withClientsOfType:

Adds the receiver as an observer of alerts when clients of the specified type sync.

```
- (void)setShouldSynchronize:(BOOL)flag withClientsOfType:(NSString *)clientType
```

Discussion

If *flag* is YES the receiver is added; otherwise the receiver is removed as an observer for alerts of the specified type. Alternatively, you can specify this information when registering the client using the client description file. You can invoke this method multiple times to register additional client types.

Typically, you use this method to setup a dependency between two clients. For example, Address Book might observe all types of clients, and is given an opportunity to join any syncs which synchronize entities defined in the contacts schema. The .Mac client might observe only device clients, so it can join a Palm or phone sync session. The client is notified only if it has entities in common with the client that initiated the sync.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [shouldSynchronizeWithClientsOfType:](#) (page 30) (`ISyncManager`)
- [setSyncAlertHandler:selector:](#) (page 28)
- [setSyncAlertToolPath:](#) (page 29)
- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38)

Related Sample Code

People

StickiesExample

Declared In

`ISyncClient.h`

setSyncAlertHandler:selector:

Sets the target and action to be invoked when an observed client creates a session and begins syncing.

```
- (void)setSyncAlertHandler:(id)handler selector:(SEL)selector
```

Discussion

When *selector* is sent to *handler*, your client has the opportunity to join the sync session.

The *selector* method is expected to take the receiver (an `ISyncClient` object) as the first argument and an array of entity names (an `NSArray` object) as the second argument. The method signature for *selector* should look like:

```
- (void)client:(ISyncClient *)client willSyncEntityNames:(NSArray *)entityNames
```

If *selector* returns without creating a session, the sync engine assumes the client will not join the session. If this client already has another handler registered—for example, from another client process—this method raises an exception. An observer is automatically removed when the client terminates.

When you create a session using the [beginSessionWithClient:entityNames:beforeDate:](#) (page 58) `ISyncSession` class method, you specify how long you are willing to wait for the sync session. This is the length of time you are willing to wait for all the other clients to join the session. If a client takes too long to join a session, the sync engine may proceed without it.

Use this method instead of [setSyncAlertToolPath:](#) (page 29) if you want to notify a running application only. Use [setShouldSynchronize:withClientsOfType:](#) (page 27) to specify the types of clients the receiver wishes to observe. If both a tool and an observer are registered, only the observer is notified.

Note: If your client is multithreaded, the thread that registers the alert handler has to exist and have a run loop running, otherwise the client does not receive the alert.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [shouldSynchronizeWithClientsOfType:](#) (page 30)
- [syncAlertToolPath](#) (page 31)

Related Sample Code

People

StickiesExample

Declared In

ISyncClient.h

setSyncAlertToolPath:

Specifies the absolute path to a tool that is launched when an observed client creates a session and begins syncing.

```
- (void)setSyncAlertToolPath:(NSString *)path
```

Discussion

The sync engine retains this path until the client is unregistered or you explicitly change the path using this method. Pass `nil` if you want to disable the sync alert tool.

When the tool is launched it passes the following command-line arguments:

```
--sync <clientIdentifier> --entitynames <entityNames>
```

The *clientIdentifier* argument is the identifier of the observed client. The *entityNames* argument is a single string containing the entity names delimited by commas that is synced. You can send `componentsSeparatedByString:` to the string with `@", "` as the argument to convert it to an array of entity names. The order of the key-value pairs, where `--sync` and `--entitynames` are keys, is arbitrary. If the tool terminates without creating a sync session, the sync engine assumes the client will not join the session.

When you create a session using the `beginSessionWithClient:entityNames:beforeDate:` (page 58) `ISyncSession` class method, you specify how long you are willing to wait for the sync session. This is the time you are willing to wait for all the other clients to join the session. If a client takes too long to join a session, the sync engine may proceed without it.

Use this method instead of `setSyncAlertHandler:selector:` (page 28) if you want to notify an application or tool that may not be running. Use `setShouldSynchronizeWithClientsOfType:` (page 27) to specify the types of clients the receiver wishes to observe. If both a tool and a handler are registered, only the handler is notified.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [shouldSynchronizeWithClientsOfType:](#) (page 30)
- [syncAlertToolPath](#) (page 31)

Declared In

ISyncClient.h

shouldReplaceClientRecordsForEntityName:

Returns YES if the client should replace all records for the entity specified by *entityName* during the next sync, NO otherwise.

- (BOOL)shouldReplaceClientRecordsForEntityName:(NSString *)*entityName*

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setShouldReplaceClientRecords:forEntityNames:](#) (page 27)

Declared In

ISyncClient.h

shouldSynchronizeWithClientsOfType:

Returns YES if the client is registered to receive alerts when clients of *clientType* sync, NO otherwise.

- (BOOL)shouldSynchronizeWithClientsOfType:(NSString *)*clientType*

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setShouldSynchronize:withClientsOfType:](#) (page 27)
- [setSyncAlertHandler:selector:](#) (page 28)
- [setSyncAlertToolPath:](#) (page 29)
- [syncAlertToolPath](#) (page 31)

Declared In

ISyncClient.h

supportedEntityNames

Returns an array of NSString objects containing the names of the entities the client supports.

- (NSArray*)supportedEntityNames

Discussion

This property is set when registering the client.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) (ISyncManager)

Declared In

ISyncClient.h

syncAlertToolPath

Returns the path to the tool that is launched when an observed client begins syncing.

- (NSString *)syncAlertToolPath

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setShouldSynchronize:withClientsOfType:](#) (page 27)

- [setSyncAlertHandler:selector:](#) (page 28)

- [setSyncAlertToolPath:](#) (page 29)

- [shouldSynchronizeWithClientsOfType:](#) (page 30)

Declared In

ISyncClient.h

Constants

Use the following constants to specify the type of client you might want to sync simultaneously with using the [setShouldSynchronize:withClientsOfType:](#) (page 27) method. The [clientType](#) (page 21) method also returns one of these constants.

Constant	Description
ISyncClientTypeApplication	Indicates the client is an application, such as Mail or iCal.
ISyncClientTypeDevice	Indicates the client is used to sync a device such as a phone or an iPod.
ISyncClientTypeServer	Indicates the client is used to sync a remote server such as .Mac.
ISyncClientTypePeer	Indicates the client is a peer, such as another computer.

The following constants are returned by the [lastSyncStatusForEntityName:](#) (page 24) method to indicate the state of the last sync session.

Constant	Description
<code>ISyncStatusRunning</code>	Indicates the client is syncing. Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .
<code>ISyncStatusSuccess</code>	Indicates the last sync was successful. Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .
<code>ISyncStatusWarnings</code>	Indicates the last sync resulted in warnings. Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .
<code>ISyncStatusErrors</code>	Indicates the last sync resulted in errors. Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .
<code>ISyncStatusCancelled</code>	Indicates the last sync was canceled. Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .
<code>ISyncStatusFailed</code>	Indicates the last sync failed to complete (for example, the client crashed). Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .
<code>ISyncStatusNever</code>	Indicates the client has never synced. Available in Mac OS X v10.4 and later. Declared in <code>ISyncClient.h</code> .

ISyncFilter Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncFilter.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide

Overview

ISyncFilter provides a set of standard filters that can be used by any client, and some utility methods for creating filters. You should never instantiate or subclass ISyncFilter directly.

If your application uses only a subset of the entities, attributes, and relationships defined in a schema, then you can restrict pulled records to that subset using filters. A filter is any object that conforms to the ISyncFiltering protocol. You set filters using the [setFilters:](#) (page 25) ISyncClient method.

Use the methods in this class to create new filters by applying logical AND and OR binary operators on a collection of filters. Use the [filterMatchingAllFilters:](#) (page 34) to apply an AND operator and [filterMatchingAtLeastOneFilter:](#) (page 34) to apply an OR operator.

See *Sync Services Programming Guide* for more information on using filters.

Tasks

Creating and Initializing Instances

- + [filterMatchingAllFilters:](#) (page 34)
Returns a filter that is the logical AND of the filters specified in the *filters* array.
- + [filterMatchingAtLeastOneFilter:](#) (page 34)
Returns a filter that is the logical OR of the filters specified in the *filters* array.

Class Methods

filterMatchingAllFilters:

Returns a filter that is the logical AND of the filters specified in the *filters* array.

```
+ (id <ISyncFiltering>)filterMatchingAllFilters:(NSArray *)filters
```

Discussion

All the filters are expected to conform to the ISyncFiltering protocol.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncFilter.h

filterMatchingAtLeastOneFilter:

Returns a filter that is the logical OR of the filters specified in the *filters* array.

```
+ (id <ISyncFiltering>)filterMatchingAtLeastOneFilter:(NSArray *)filters
```

Discussion

All the filters are expected to conform to the ISyncFiltering protocol.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncFilter.h

ISyncManager Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncManager.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide
Related sample code	People SeeMyFriends StickiesExample

Overview

You use an `ISyncManager` object to communicate directly with the sync engine to perform administrative operations. A client must register itself with an `ISyncManager` object before it can sync its data. If a client is not using an existing schema, it must register the schema before it registers itself. You also use an `ISyncManager` to look up an existing client and unregister a client.

There's only one `ISyncManager` instance per client process you obtain using the [sharedManager](#) (page 37) class method. You should never instantiate or subclass `ISyncManager` directly.

`Sync Services` provides three canonical schemas: `Bookmarks.syncschema`, `Contacts.syncschema`, and `Calendars.syncschema`. If you want to extend one of these existing schemas or define your own schema, then you need to register that schema with the shared `ISyncManager` object. You use the [registerSchemaWithBundlePath:](#) (page 39) method to register a schema with the sync engine or update an existing schema. Occasionally, you might use [unregisterSchemaWithName:](#) (page 41) to remove a schema and all associated records. Removing a schema impacts every client that uses that schema. Typically, you just register a schema once and reregister it when it changes.

You also use the shared `ISyncManager` object to create and register a sync client—that is, an instance of `ISyncClient`—with a unique identifier that you specify. Use [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) to register your client—this method returns either a new client object or an existing client. You also use this method to describe the capabilities of the client—for example, describe what entities and properties the client supports. You use the [unregisterClient:](#) (page 41) method to unregister a client. See *Sync Services Programming Guide* for more information on registering and unregistering clients.

Tasks

Getting the Default Manager

- + [sharedManager](#) (page 37)
Returns a shared ISyncManager object.

Getting a Manager's State

- [isEnabled](#) (page 38)
Returns NO if the sync engine is disabled, YES otherwise.

Registering Schemas

- [registerSchemaWithBundlePath](#): (page 39)
Registers a schema property list located in a bundle at *bundlePath*.
- [unregisterSchemaWithName](#): (page 41)
Unregisters a schema uniquely identified by *schemaName*, and removes all associated records.

Registering Clients

- [clientWithIdentifier](#): (page 38)
Returns the sync client identified by *clientIdentifier*, or nil if not found.
- [registerClientWithIdentifier:descriptionFilePath](#): (page 38)
Returns an existing or new sync client uniquely identified by *clientIdentifier*.
- [unregisterClient](#): (page 41)
Unregisters a sync client represented by *client*.

Getting Snapshots

- [snapshotOfRecordsInTruthWithEntityNames:usingIdentifiersForClient](#): (page 40)
Returns an immutable snapshot of the records for *entityNames* from the truth database.

Using Sync Alert Handlers

- [addRequestMode](#): (page 37)
Adds a mode to the set of run-loop input modes that the receiver uses for connection requests.
- [removeRequestMode](#): (page 40)
Removes a mode from the set of run-loop input modes the receiver uses for connection requests.
- [requestModes](#) (page 40)
Returns the set of request modes the receiver registers with its NSRunLoop object.

Class Methods

sharedManager

Returns a shared ISyncManager object.

```
+ (ISyncManager *)sharedManager
```

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

People

SeeMyFriends

StickiesExample

Declared In

ISyncManager.h

Instance Methods

addRequestMode:

Adds a mode to the set of run-loop input modes that the receiver uses for connection requests.

```
- (void)addRequestMode:(NSString *)mode
```

Parameters

mode

The mode to add to the receiver. See *NSRunLoop Class Reference* for more information on input modes.

Discussion

Clients that register sync alert handlers may use this method to manage the request modes of connections that are sent alerts by the sync engine. This method is similar to the `addRequestMode:` method of `NSConnection`. For example, a client that registers a sync alert handler in a process that might present a modal dialog to the user, should add the appropriate request mode to the run-loop, so alerts can be handled in a timely manner even when the application is blocked for user input.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [removeRequestMode:](#) (page 40)

- [requestModes](#) (page 40)

- `addRequestMode:` (`NSConnection`)

Declared In

ISyncManager.h

clientWithIdentifier:

Returns the sync client identified by *clientIdentifier*, or nil if not found.

```
- (ISyncClient *)clientWithIdentifier:(NSString *)clientIdentifier
```

Availability

Available in Mac OS X v10.4 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38)
- [unregisterClient:](#) (page 41)

Related Sample Code

SeeMyFriends

StickiesExample

Declared In

ISyncManager.h

isEnabled

Returns NO if the sync engine is disabled, YES otherwise.

```
- (BOOL)isEnabled
```

Discussion

You should not begin a sync session when this method returns NO. However, you can register for the [ISyncAvailabilityChangedNotification](#) (page 42) notification, which is sent when the sync engine state changes.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncManager.h

registerClientWithIdentifier:descriptionFilePath:

Returns an existing or new sync client uniquely identified by *clientIdentifier*.

```
- (ISyncClient *)registerClientWithIdentifier:(NSString *)clientIdentifier
descriptionFilePath:(NSString *)descriptionFilePath
```

Discussion

There are no restrictions on the content or length of *clientIdentifier*, but it must be unique across all clients. Typically, it's a DNS-style name such as `com.apple.iCal`.

The client description file located at *descriptionFilePath* is a property list that specifies client information that the sync engine needs to know to sync its records. For example, the client description file a list of the client supported entities and properties. See *Sync Services Programming Guide* for a complete description of the client description file.

If the client already exists, then invoking this method updates the client description. If the set of supported entities and properties changes, the sync engine may force the client to slow sync the next time it syncs. This can be expensive, so only reregister a client if necessary.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [clientWithIdentifier:](#) (page 38)
- [unregisterClient:](#) (page 41)
- [canPullChangesForEntityName:](#) (page 20) (ISyncClient)
- [canPushChangesForEntityName:](#) (page 20) (ISyncClient)
- [displayName:](#) (page 21) (ISyncClient)
- [imagePath:](#) (page 22) (ISyncClient)

Related Sample Code

People

SeeMyFriends

StickiesExample

Declared In

ISyncManager.h

registerSchemaWithBundlePath:

Registers a schema property list located in a bundle at *bundlePath*.

- (BOOL)registerSchemaWithBundlePath:(NSString *)*bundlePath*

Discussion

The schema can define new entities and properties, and extend existing entities. The schema bundle may contain other files, such as images and localization files. See *Sync Services Programming Guide* for more details on the schema format and contents of the schema bundle.

If a schema of the same name exists, invoking this method updates that schema. Consequently, records and properties of records may be removed if an entity or property is removed from the schema. This action may cause clients that use this schema to slow sync the next time they sync. This process can be expensive, so reregister a schema only if necessary.

Returns YES if successful, NO otherwise.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [unregisterSchemaWithName:](#) (page 41)

Related Sample Code

StickiesExample

Declared In

ISyncManager.h

removeRequestMode:

Removes a mode from the set of run-loop input modes the receiver uses for connection requests.

```
(void)removeRequestMode:(NSString *)mode
```

Parameters

mode

The mode to remove. See *NSRunLoop Class Reference* for more information on input modes.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [addRequestMode:](#) (page 37)
- [requestModes](#) (page 40)
- [removeRequestMode:](#) (NSConnection)

Declared In

ISyncManager.h

requestModes

Returns the set of request modes the receiver registers with its NSRunLoop object.

```
(NSArray *)requestModes
```

Return Value

An array of NSString objects that represents the set of request modes that the receiver registers. See *NSRunLoop Class Reference* for more information on input modes.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [addRequestMode:](#) (page 37)
- [removeRequestMode:](#) (page 40)
- [requestModes](#) (NSConnection)

Declared In

ISyncManager.h

snapshotOfRecordsInTruthWithEntityNames:usingIdentifiersForClient:

Returns an immutable snapshot of the records for *entityNames* from the truth database.

```
-(ISyncRecordSnapshot *)snapshotOfRecordsInTruthWithEntityNames:(NSArray *)entityNames usingIdentifiersForClient:(ISyncClient *)client
```


Discussion

The truth database stores a copy of all the synced records and contains the amalgamation of all entities and properties from all clients. The snapshot is made of the records for entities specified by the *entityNames* argument, an array of NSString objects containing the names of entities. You access the records by sending messages to the returned ISyncRecordSnapshot object.

Each client has its own name space for record identifiers. The *client* argument specifies the name space you want to use. If *client* is nil or invalid, the record identifiers from the sync engine's global name space are used.

The snapshot is an immutable copy of the records taken at the time returned object is created. If the truth database is subsequently modified, the changes are not be reflected in the snapshot. You should create a new snapshot if you want up-to-date records.

Do not use this method if you are syncing and want a snapshot that is consistent with the sync session. Another client may be pushing changes that you have not pulled yet. Instead, you can use the ISyncSession [snapshotOfRecordsInTruth](#) (page 75) method to get the state of a session.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [clientWithIdentifier:](#) (page 38)

Related Sample Code

SeeMyFriends

Declared In

ISyncManager.h

unregisterClient:

Unregisters a sync client represented by *client*.

```
- (void)unregisterClient:(ISyncClient *)client
```

Discussion

Does nothing if *client* is not registered.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [clientWithIdentifier:](#) (page 38)

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38)

Declared In

ISyncManager.h

unregisterSchemaWithName:

Unregisters a schema uniquely identified by *schemaName*, and removes all associated records.

- (void)unregisterSchemaWithName:(NSString *)*schemaName*

Discussion

This action causes clients that use this schema to slow sync the next time they sync. This can be expensive and results in the loss of data, so only unregister a schema if necessary. This method does nothing if the schema is not registered.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [registerSchemaWithBundlePath](#): (page 39)

Declared In

ISyncManager.h

Constants

The following constant is thrown whenever the connection with the sync engine is lost.

Constant	Description
ISyncServer- UnavailableException	A string aggregating the name, reason, and user info from the originating exception. Thrown by any ISyncManager method when communication to the server is lost.

Notifications

ISyncAvailabilityChangedNotification

Posted by the distributed notification center when syncing is enabled or disabled. The notification object is an NSString equal to "YES" if enabled and "NO" if disabled. The receiver should still invoke [isEnabled](#) (page 38) before beginning a sync session. This notification does not contain a `userInfo` dictionary.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncManager.h

ISyncRecordReference Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncRecordReference.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Sync Services Programming Guide

Overview

An `ISyncRecordReference` object provides a reusable and optionally persistent representation of a Sync Services record. For various reasons, a record identifier may change over time—for example, a record might be replaced during a refresh sync. Therefore, a record identifier might change when a record is logically the same. Alternatively, an `ISyncRecordReference` object provides a reliable reference to a Sync Services record that can be archived and shared among processes.

You do not create `ISyncRecordReference` objects directly. Instead, you use methods of the `ISyncRecordSnapshot` class, as follows: Use the `recordReferenceForRecordWithIdentifier:` (page 47) method to get the `ISyncRecordReference` object associated with a given record identifier. Use the `recordIdentifierForReference:isModified:` (page 46) method to get the record identifier associated with a given `ISyncRecordSnapshot` object. The record identifiers are in the scope of the associated record snapshot.

An `ISyncRecordReference` object conforms to the `NSCoding` protocol, and therefore can be archived and unarchived. Typically, clients archive record reference objects after a sync and unarchive them before a sync.

ISyncRecordSnapshot Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncRecordSnapshot.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide
Related sample code	SeeMyFriends

Overview

The `ISyncRecordSnapshot` class provides a client with access to an immutable copy of the records in the truth database. The truth database is an aggregate of all records of all supported entities and properties of all registered clients. `ISyncRecordSnapshot` provides a variety of methods for querying the records in the snapshot.

There is no mutable variant of `ISyncRecordSnapshot` for changing records—only the sync engine updates the truth by clients pushing records during a sync session. If the sync engine modifies records after a snapshot is created, the snapshot does not contain those changes. You must create a new snapshot to get the latest records.

Use the [snapshotOfRecordsInTruthWithEntityNames:usingIdentifiersForClient:](#) (page 40) `ISyncManager` method to create a snapshot of the truth database. Use the [recordsWithIdentifiers:](#) (page 48) method to get a dictionary representation of all the specified records where the keys in the returned dictionary are the record identifiers. For example, a configuration tool might create a snapshot to query records in the truth database and provide various filtering.

However, if you are syncing and want a snapshot that is consistent with the sync session (for example, another client is pushing changes that you have not pulled yet), then you need to use the [snapshotOfRecordsInTruth](#) (page 75) `ISyncSession` method to get a snapshot.

Other methods in this class are used to query the snapshot. Use the [recordsWithIdentifiers:](#) (page 48) method to get specific records (it is more efficient to request records in batches, not all at once). Use the [recordsWithMatchingAttributes:](#) (page 48) method to get records that match specific attribute values, and the [sourceIdentifiersForRelationshipName:withTargetIdentifier:](#) (page 48) or [targetIdentifiersForRelationshipName:withSourceIdentifier:](#) (page 49) method to get records that match relationship values. Because all of these query methods can be computationally intensive, use them with care.

You can also use an `ISyncRecordReference` object to maintain a persistent reference to a record. Use the `recordReferenceForRecordWithIdentifier:` (page 47) method to get an `ISyncRecordReference` object.

Tasks

Getting Records

- `recordsWithIdentifiers:` (page 48)
Returns a dictionary containing the records.

Getting Records in a Relationship

- `targetIdentifiersForRelationshipName:withSourceIdentifier:` (page 49)
Returns an array of record identifiers belonging to the target objects of a relationship.
- `sourceIdentifiersForRelationshipName:withTargetIdentifier:` (page 48)
Returns an array of the record identifiers belonging to the source objects of a relationship.

Searching for Records

- `recordsWithMatchingAttributes:` (page 48)
Returns a dictionary containing all the records that match a query.

Getting Record References

- `recordReferenceForRecordWithIdentifier:` (page 47)
Returns a record reference that corresponds to the given record identifier.
- `recordIdentifierForReference:isModified:` (page 46)
Returns the record identifier that corresponds to the given record reference object.

Instance Methods

recordIdentifierForReference:isModified:

Returns the record identifier that corresponds to the given record reference object.

- `(NSString *)recordIdentifierForReference:(ISyncRecordReference *)reference isModified:(BOOL *)modifyFlag`

Parameters*reference*

Specifies the record reference object corresponding to the returned record identifier. This method may change internal properties of this object. Use the *modifyFlag* argument to determine if this object changes.

modifyFlag

Optional flag used to determine if this method changes any properties of *reference*. For example, if internal properties change since *reference* was created or since this method was last invoked, this method updates *reference* and sets *modifyFlag* accordingly. If *modifyFlag* is non-NULL and *reference* changes then *modifyFlag* is set to YES. Otherwise, if *modifyFlag* is non-NULL then it is set to NO.

Return Value

Returns the record identifier corresponding to the specified ISyncRecordReference *reference* argument. The returned identifier is in the scope of the receiver. For example, if the receiver is a global snapshot, then the returned record identifier is a global identifier. Returns `nil` if no record exists that corresponds to *reference*.

Discussion

Typically, a client archives record reference objects and unarchives them before syncing. The client uses this method to get the local record identifiers for the archived record references. If the original record is not present in the truth database then this method returns the record identifier for the record that matches the identity properties. If more than one record matches the identity properties, then this method returns the record identifier of the first record found.

If the record reference was unarchived before invoking this method, it may be stale and need to be modified internally. In this case, use the *modifyFlag* argument in this method and if the record reference changes, save it again. This insures that the most current record reference objects are used in subsequent syncs.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncRecordSnapshot.h

recordReferenceForRecordWithIdentifier:

Returns a record reference that corresponds to the given record identifier.

```
- (ISyncRecordReference *)recordReferenceForRecordWithIdentifier:(NSString *)identifier
```

Parameters*identifier*

A record identifier that is in the scope of the receiver. For example, if the receiver was created using the [snapshotOfRecordsInTruth](#) (page 75) ISyncSession method, then *identifier* must be a global identifier. If the receiver was created using the [snapshotOfRecordsInTruthWithEntityNames:usingIdentifiersForClient:](#) (page 40) ISyncManager method, then *identifier* must be in the scope of the client.

Return Value

Returns the record reference associated with *identifier*. Returns `nil` if no record can be found for *identifier*.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncRecordSnapshot.h

recordsWithIdentifiers:

Returns a dictionary containing the records.

```
- (NSDictionary *)recordsWithIdentifiers:(NSArray *)recordIdentifiers
```

Discussion

Returns a dictionary containing the records specified by *recordIdentifiers*, an array of record identifiers. The dictionary keys are the record identifiers, and the values are the record dictionaries. The returned dictionary does not contain key-value pairs for deleted records or invalid record identifiers.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncRecordSnapshot.h

recordsWithMatchingAttributes:

Returns a dictionary containing all the records that match a query.

```
- (NSDictionary *)recordsWithMatchingAttributes:(NSDictionary *)attributes
```

Discussion

The *attributes* argument specifies the query and supports key-value pairs for record properties. For example, a key might be `firstName` and a value might be "Jane" in the *attributes* dictionary. Optionally, you can use the `ISyncRecordEntityNameKey` key to limit the search to a particular entity. The keys in the returned dictionary are the record identifiers, and the values are the record dictionaries that match the query.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncRecordSnapshot.h

sourceIdentifiersForRelationshipName:withTargetIdentifier:

Returns an array of the record identifiers belonging to the source objects of a relationship.

```
- (NSArray *)sourceIdentifiersForRelationshipName:(NSString *)relationshipName
withTargetIdentifier:(NSString *)targetIdentifier
```

Discussion

Returns an array of the record identifiers of the source objects belonging to the relationship, specified by *relationshipName*, whose target object identifier is *targetIdentifier*. You can use this method to get the inverse of a to-many or to-one relationship when that relationship is not defined in the schema.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [targetIdentifiersForRelationshipName:withSourceIdentifier:](#) (page 49)

Declared In

ISyncRecordSnapshot.h

targetIdentifiersForRelationshipName:withSourceIdentifier:

Returns an array of record identifiers belonging to the target objects of a relationship.

```
- (NSArray *)targetIdentifiersForRelationshipName:(NSString *)relationshipName
    withSourceIdentifier:(NSString *)sourceIdentifier
```

Discussion

Returns an array of the record identifiers of the target objects belonging to the relationship, specified by *relationshipName*, whose source object identifier is *sourceIdentifier*. If *relationshipName* is a to-one relationship, then the returned array contains no more than one object, otherwise it may contain multiple objects. If the relationship is ordered, the order of the record identifiers matches the order of the records. A record identifier may appear in the array multiple times. Use this method if you want to resolve the destination objects of a relationship.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [sourceIdentifiersForRelationshipName:withTargetIdentifier:](#) (page 48)

Declared In

ISyncRecordSnapshot.h

ISyncSession Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncSession.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide
Related sample code	People SeeMyFriends StickiesExample

Overview

An `ISyncSession` object is used to manage a single sync operation. It coordinates communication between a client, the sync engine, and any other clients that sync simultaneously. You create an `ISyncSession` object when you want to sync—you use it to sync your records and then throw it away.

A sync session is modeled as a finite state machine whose four main states are negotiation, pushing, mingling, and pulling. During the negotiation state, the client requests a sync mode, such as **slow sync**, **fast sync**, or **pull the truth**. The client then pushes changes to the sync engine. When all changes from all participating clients are pushed, the sync engine enters the mingling state. During mingling, it processes all the pushed records and computes the changes that are pulled by each client. After mingling, the client pulls changes from the sync engine.

You should use sync anchors so your application is more resilient by avoiding serious errors that can occur during a sync session—such as a communication failure between a client and its data store that corrupts data. Use sync anchors unless you implement your own mechanism for tracking whether records are successfully pushed and pulled. A sync anchor is an object that is unique per client and per entity that is saved periodically throughout a sync session. The sync engine compares the client's locally stored sync anchors with its copies to determine the next sync mode. For example, if there is a discrepancy in sync anchors and the client is an application, an alert panel appears asking the user to select an appropriate sync mode. If the client is not an application, it slow syncs.

The client can finish or cancel a session at any time. Read *Sync Services Programming Guide* for more details about each state in the finite state machine, transactions within each state, and the consequences of invoking `ISyncSession` methods. Refer to *Sync Services Programming Guide* for definitions of other sync terms.

Use the `beginSessionWithClient:entityNames:beforeDate:lastAnchors:` (page 59) method to create an `ISyncSession` object for the specified client and entities. Alternatively, use the `beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:` (page 56) method if you don't want to block when creating a session. Creating a session might block if the sync engine is waiting for other clients to join the sync.

During the negotiation state, use the `clientWantsToPushAllRecordsForEntityNames:` (page 68) or the `clientDidResetEntityNames:` (page 65) method to request a sync mode different from the default mode.

During the pushing state, use the `pushChange:` (page 71) method to push just the changes for a particular record. Otherwise, use the `pushChangesFromRecord:withIdentifier:` (page 72) method to push the entire record and let the sync engine figure out which properties changed. Use the `deleteRecordWithIdentifier:` (page 68) method to push a delete change. After successfully pushing your entities, use the `clientFinishedPushingChangesWithNextAnchors:` (page 66) method to change the sync anchors stored with the sync engine.

Use the `prepareToPullChanges...` methods to begin the mingling state and transition to the pulling state. During the mingling state, the sync engine merges all the changes between multiple clients and computes what changes need to be pulled by clients. Hence, these methods prepare the sync session for the pulling state.

During the pulling state, use the `changeEnumeratorForEntityNames:` (page 61) method to pull all the changes for the specified entities. Use the returned object enumerator to iterate through and apply the changes to your data. Use the `clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:` (page 62) method to accept changes and the `clientRefusedChangesForRecordWithIdentifier:` (page 67) method to reject changes. Use the `clientCommittedAcceptedChangesWithNextAnchors:` (page 64) method to commit the accepted changes and close a transaction within the pulling state.

Use `finishSyncing` (page 69) to terminate a sync session by closing an open transaction. Or use `cancelSyncing` (page 61) to cancel a sync session that rolls back the state of the client to the previously closed transaction. All changes that were applied in an open transaction need to be reapplied on the next sync. The `finishSyncing` (page 69) and `cancelSyncing` (page 61) methods can be invoked at any time. However, the `ISyncSession` object should be released after invoking these methods, because the object cannot be used in a subsequent sync.

If you are using Mac OS X v10.4 or earlier, use the `beginSessionWithClient:entityNames:beforeDate:` (page 58) method to create an `ISyncSession` object for the specified client and entities. Alternatively, use the `beginSessionInBackgroundWithClient:entityNames:target:selector:` (page 55) method if you don't want to block when creating a session. Similarly, use the non-anchor `clientCommittedAcceptedChanges` (page 64) method to commit the accepted changes and close a transaction within the pulling state.

Tasks

Creating a Sync Session

- + [beginSessionWithClient:entityNames:beforeDate:lastAnchors:](#) (page 59)
Creates and returns a new sync session for the specified client using sync anchors.
- + [beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:](#) (page 56)
Creates a new sync session for the specified client asynchronously using sync anchors.
- + [beginSessionWithClient:entityNames:beforeDate:](#) (page 58)
Creates and returns a new sync session for the specified client.
- + [beginSessionInBackgroundWithClient:entityNames:target:selector:](#) (page 55)
Creates a new sync session for the specified client asynchronously.
- + [cancelPreviousBeginSessionWithClient:](#) (page 60)
Cancels a previous request to create a session using [beginSessionInBackgroundWithClient:entityNames:target:selector:](#) (page 55) for *client*.

Negotiating a Sync Mode

- [clientDidResetEntityNames:](#) (page 65)
Tells the sync engine to perform a refresh sync of all the records for the specified entities.
- [clientWantsToPushAllRecordsForEntityNames:](#) (page 68)
Forces a slow sync of all the records for the specified entities.
- [shouldPushChangesForEntityName:](#) (page 74)
Returns YES if the client should push changes to records for *entityName* since the last sync, NO otherwise.
- [shouldPushAllRecordsForEntityName:](#) (page 73)
Returns YES if the client should push all the records for *entityName* to the sync engine; otherwise, NO.
- [shouldPullChangesForEntityName:](#) (page 73)
Returns YES if the client should pull changes to records for *entityName*, NO otherwise.
- [shouldReplaceAllRecordsOnClientForEntityName:](#) (page 74)
Returns YES if the client should delete all the records for the entity, specified by *entityName*, and replace them with records pulled from the sync engine, NO otherwise.

Pushing Changes

- [pushChange:](#) (page 71)
Pushes changes made to a single record, specified by *change*, to the sync engine.
- [pushChangesFromRecord:withIdentifier:](#) (page 72)
Compares *record* to the client's previous known state of the record, identified by *recordIdentifier*, and pushes the changes to the sync engine.

- [deleteRecordWithIdentifier:](#) (page 68)
Creates a delete change for the record specified by *recordIdentifier* and pushes the change to the sync engine.
- [clientFinishedPushingChangesWithNextAnchors:](#) (page 66)
Sets the sync anchors for each entity whose records were successfully pushed by the client.

Mingling

- [prepareToPullChangesForEntityNames:beforeDate:](#) (page 70)
Moves the receiver to the mingling state and returns when the sync engine is ready for the client to begin pulling changes to the specified entities.
- [prepareToPullChangesInBackgroundForEntityNames:target:selector:](#) (page 71)
Moves the receiver to the mingling state and sends a message to a specified target when the sync engine is ready for the client to begin pulling changes to the specified entities.

Pulling Changes

- [changeEnumeratorForEntityNames:](#) (page 61)
Returns the object enumerator for the ISyncChange objects which contain all the changes the client should apply to its local data.
- [clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:](#) (page 62)
Informs the sync engine that the client has accepted the changes to the record identified by *recordIdentifier* during the pulling state.
- [clientRefusedChangesForRecordWithIdentifier:](#) (page 67)
Informs the sync engine during the pulling state that the client has refused to apply the changes for the record specified by *recordIdentifier*.
- [clientCommittedAcceptedChanges](#) (page 64)
Informs the sync engine that all accepted and rejected changes in the current transaction during the pulling state should be committed.
- [clientCommittedAcceptedChangesWithNextAnchors:](#) (page 64)
Sets the sync anchors for each entity whose records were successfully updated during the pulling phase.
- [clientChangedRecordIdentifiers:](#) (page 63)
Changes the record identifiers of the given records.

Pushing and Pulling Changes

- [clientLostRecordWithIdentifier:shouldReplaceOnNextSync:](#) (page 67)
Tells the sync engine that a record identified by *recordIdentifier*, no longer exists on the client, and indicates whether or not it should be replaced.

Finishing Syncing

- [finishSyncing](#) (page 69)
Tells the sync engine that the client is done syncing. Invoking this method closes any open transactions in the pushing or pulling states.

Canceling Syncing

- [isCancelled](#) (page 69)
Returns YES if the receiver was canceled, NO otherwise.
- [cancelSyncing](#) (page 61)
Cancels the current session.

Getting and Setting Client Information

- [clientInfoForRecordWithIdentifier:](#) (page 66)
Returns a client-specific, nonprepsynchronized object that stores additional information about a record specified by *recordIdentifier*.
- [setClientInfo:forRecordWithIdentifier:](#) (page 73)
Associates a client-specific, non synchronized object, *clientInfo*, to a record specified by *recordIdentifier*.

Getting Snapshots

- [snapshotOfRecordsInTruth](#) (page 75)
Returns an immutable snapshot of the records in the truth database.

Class Methods

beginSessionInBackgroundWithClient:entityNames:target:selector:

Creates a new sync session for the specified client asynchronously.

```
+ (void)beginSessionInBackgroundWithClient:(ISyncClient *)client entityNames:(NSArray *)entityNames target:(id)target selector:(SEL)selector
```

Parameters

client

The client that is syncing.

entityNames

An array of entity names that the client wants to sync.

The *entityNames* parameter can be a subset of the client's supported entities and may include entities that have been disabled. However, the sync engine does not allow the client to push changes to disabled entities nor does it provide changes to disabled entities while pulling changes. Typically, you use the array returned by sending `enabledEntityNames` (page 22) to your `ISyncClient` object as the *entityNames* parameter to this method.

target

The recipient of *selector*.

selector

The message to send to *target*.

The *selector* message is passed two parameters: The first parameter is the `ISyncClient` object and the second parameter is the new `ISyncSession` object. If the sync engine is disabled or another client already created a session for this client, then this method fails to create a session and *selector* is sent to *target* with `nil` as the `ISyncSession` object.

Discussion

Creating a session for *client* may trigger notifications to other clients observing syncs of this client type. This method differs from `beginSessionWithClient:entityNames:beforeDate:` (page 58) by not blocking—returning immediately—and sending *selector* to *target* when all dependent clients have joined the sync session. (Send `setShouldSynchronize:withClientsOfType:` (page 27) to an `ISyncClient` object to setup these dependencies.) This method requires the client have a run loop running in the default mode.

Note: `ISyncSession` is not thread-safe. You can pass an `ISyncSession` object between threads but you should not use it concurrently. Asynchronous callbacks from `ISyncSession` are delivered to any client thread that used any of the Sync Services methods. The client is responsible for directing the callback to an appropriate thread.

Availability

Available in Mac OS X v10.4 and later.

See Also

- + `cancelPreviousBeginSessionWithClient:` (page 60)
- `isEnabled` (page 38) (`ISyncManager`)

Related Sample Code

People

SeeMyFriends

Declared In

`ISyncSession.h`

`beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:`

Creates a new sync session for the specified client asynchronously using sync anchors.

```
+ (void)beginSessionInBackgroundWithClient:(ISyncClient *)client entityNames:(NSArray
    *)entityNames target:(id)target selector:(SEL)selector lastAnchors:(NSDictionary
    *)anchors
```


Parameters

client

The client that is syncing.

entityNames

An array of entity names that the client wants to sync.

The *entityNames* parameter can be a subset of the client's supported entities and may include entities that have been disabled. However, the sync engine does not allow the client to push changes to disabled entities nor does it provide changes to disabled entities while pulling changes. Typically, you use the array returned by sending [enabledEntityNames](#) (page 22) to your `ISyncClient` object as the *entityNames* parameter to this method.

target

The recipient of *selector*.

selector

The message to send to *target*.

The *selector* message is passed two parameters: the first parameter is the `ISyncClient` object and the second parameter is the new `ISyncSession` object. If the sync engine is disabled or another client already created a session for this client, then this method fails to create a session and *selector* is sent to *target* with `nil` as the `ISyncSession` object.

anchors

The sync anchors used in the last sync which are compared to those saved by the sync engine to determine the sync mode.

The keys are the entity names and the values are the sync anchors. Sync anchors are globally unique `NSString` objects, typically containing a UUID or date. If this parameter is `nil`, the client refresh syncs.

As a convenience, you may use the same sync anchor for all entities of a data class by specifying a sync anchor for only one entity in that data class. If you provide sync anchors for two or more entities per data class, then you need to specify sync anchors for all entities in that data class. A missing sync anchor for an entity causes that entity to refresh sync.

Discussion

This method is the nonblocking variation of the

[beginSessionWithClient:entityNames:beforeDate:lastAnchors:](#) (page 59) method similar to the nonblocking [beginSessionInBackgroundWithClient:entityNames:target:selector:](#) (page 55) method that doesn't use sync anchors. Read the [beginSessionWithClient:entityNames:beforeDate:lastAnchors:](#) (page 59) description for more details.

Note: `ISyncSession` is not thread-safe. You can pass an `ISyncSession` object between threads but you should not use it concurrently. Asynchronous callbacks from `ISyncSession` are delivered to any client thread that used any of the Sync Services methods. The client is responsible for directing the callback to an appropriate thread.

Availability

Available in Mac OS X v10.5 and later.

See Also

- + [beginSessionWithClient:entityNames:beforeDate:lastAnchors:](#) (page 59)
- [clientFinishedPushingChangesWithNextAnchors:](#) (page 66)
- [clientCommittedAcceptedChangesWithNextAnchors:](#) (page 64)

Declared In

ISyncSession.h

beginSessionWithClient:entityNames:beforeDate:

Creates and returns a new sync session for the specified client.

```
+ (ISyncSession *)beginSessionWithClient:(ISyncClient *)client entityNames:(NSArray *)entityNames beforeDate:(NSDate *)date
```

Parameters*client*

The client that is syncing.

entityNames

An array of entity names that the client wants to sync.

The *entityNames* parameter can be a subset of the client's supported entities and may include entities that have been disabled. However, the sync engine does not allow the client to push changes to disabled entities nor does it provide changes to disabled entities while pulling changes. Typically, you use the array returned by sending [enabledEntityNames](#) (page 22) to your `ISyncClient` object as the *entityNames* parameter to this method.

date

How long the client is willing to wait for other clients to join this session. If *date* is in the distant future, this method blocks until all dependent clients have joined the session or *date* has expired. If *date* is the current date or a past date, this method returns `nil` if the session cannot be created immediately.

Choose a future date carefully before invoking this method. If *date* is too small, dependent clients may be excluded from joining the sync. Typically, a client specifies the longest delay possible. However, if you sync before terminating an application, you might specify a zero delay by passing `[NSDate date]`.

Return ValueReturns the new sync session or `nil` if the session cannot be created immediately.

This method returns when all dependent clients have had the opportunity to join the sync session or *date* has expired, whichever occurs first. This method might block if another client is syncing an entity specified in *entityNames*. Returns `nil` if the sync engine is disabled.

Discussion

Creating a session for *client* may trigger notifications to other clients observing syncs of this client type. Other clients then have the opportunity to join this sync session; therefore, this method may block. Send [setShouldSynchronize:withClientsOfType:](#) (page 27) to an `ISyncClient` object to setup these dependencies. Use the [beginSessionInBackgroundWithClient:entityNames:target:selector:](#) (page 55) method if you don't want to block when creating a session.

Availability

Available in Mac OS X v10.4 and later.

See Also

- + [cancelPreviousBeginSessionWithClient:](#) (page 60)
- [isEnabled](#) (page 38) (`ISyncManager`)

Related Sample Code

StickiesExample

Declared In

ISyncSession.h

beginSessionWithClient:entityNames:beforeDate:lastAnchors:

Creates and returns a new sync session for the specified client using sync anchors.

```
+ (ISyncSession *)beginSessionWithClient:(ISyncClient *)client entityNames:(NSArray *)entityNames beforeDate:(NSDate *)date lastAnchors:(NSDictionary *)anchors
```

Parameters*client*

The client that is syncing.

entityNames

An array of entity names that the client wants to sync.

The *entityNames* parameter can be a subset of the client's supported entities and may include entities that have been disabled. However, the sync engine does not allow the client to push changes to disabled entities nor does it provide changes to disabled entities while pulling changes. Typically, you use the array returned by sending [enabledEntityNames](#) (page 22) to your `ISyncClient` object as the *entityNames* parameter to this method.

date

How long the client is willing to wait for other clients to join this session. If *date* is in the distant future, this method blocks until all dependent clients have joined the session or *date* has expired. If *date* is the current date or a past date, this method returns `nil` if the session cannot be created immediately.

Choose a future date carefully before invoking this method. If *date* is too small, dependent clients may be excluded from joining the sync. Typically, a client specifies the longest delay possible. However, if you sync before terminating an application, you might specify a zero delay by passing `[NSDate date]`.

anchors

Specifies the sync anchors used in the last sync which are compared to those saved by the sync engine to determine the sync mode.

The keys are the entity names and the values are the sync anchors. Sync anchors are globally unique `NSString` objects typically, containing a UUID or date. If this parameter is `nil`, the client refresh syncs.

As a convenience, you may use the same sync anchor for all entities of a data class by specifying a sync anchor for only one entity in that data class. If you provide sync anchors for two or more entities per data class, then you need to specify sync anchors for all entities in that data class. A missing sync anchor for an entity causes that entity to refresh sync.

Return ValueReturns the new sync session or `nil` if the session cannot be created immediately.

This method returns when all dependent clients have had the opportunity to join the sync session or *date* has expired, whichever occurs first. This method might block if another client is syncing an entity specified in *entityNames*. Returns `nil` if the sync engine is disabled.

Discussion

Use this method instead of `beginSessionWithClient:entityNames:beforeDate:` (page 58) if you are using sync anchors to improve the selection of a sync mode—that is, fast sync when possible and avoid unnecessary slow syncs. Use the `beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:` (page 56) method if you don't want to block when creating a session.

If the client is syncing an entity for the first time, the sync anchor for the entity should be `[NSNumber numberWithInt:0]` which causes a refresh sync. A null value for a sync anchor can also indicate that local records for the corresponding entity were removed or lost, and consequently, the client needs to refresh sync.

Otherwise, the sync anchors parameter should be the sync anchors you saved locally and passed to either the `clientFinishedPushingChangesWithNextAnchors:` (page 66) or `clientCommittedAcceptedChangesWithNextAnchors:` (page 64) methods during the last sync session.

The sync anchors you provide in this method are compared to the sync anchors from the last sync session to determine the sync mode. If a sync anchor doesn't match a sync anchor from the previous sync session and the client is an application, an alert panel appears asking the user to select an appropriate sync mode. If the user selects slow sync or the client is not an application, then the `shouldPushAllRecordsForEntityName:` (page 73) method returns YES for all entities in a data class corresponding to that sync anchor.

If you use this method you must also use the `clientFinishedPushingChangesWithNextAnchors:` (page 66) and `clientCommittedAcceptedChangesWithNextAnchors:` (page 64) methods to set sync anchors at the end of the pushing and pulling phases of the sync session. Otherwise, the sync engine assumes an error occurred and the client refresh syncs.

Availability

Available in Mac OS X v10.5 and later.

See Also

- + `beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:` (page 56)
- `clientFinishedPushingChangesWithNextAnchors:` (page 66)
- `clientCommittedAcceptedChangesWithNextAnchors:` (page 64)

Declared In

`ISyncSession.h`

cancelPreviousBeginSessionWithClient:

Cancels a previous request to create a session using

`beginSessionInBackgroundWithClient:entityNames:target:selector:` (page 55) for *client*.

```
+ (void)cancelPreviousBeginSessionWithClient:(ISyncClient *)client
```

Discussion

Use the `beginSessionWithClient:entityNames:beforeDate:` (page 58) method if you prefer to block when creating a session.

Availability

Available in Mac OS X v10.4 and later.

See Also

+ [beginTransactionInBackgroundWithClient:entityNames:target:selector:](#) (page 55)

+ [beginTransactionWithClient:entityNames:beforeDate:](#) (page 58)

Declared In

ISyncSession.h

Instance Methods

cancelSyncing

Cancels the current session.

- (void)cancelSyncing

Discussion

May close an open pull or push transaction, and rolls back the state of the client to the previous transaction.

In the case of a pull transaction, the sync engine assumes the client is able to reapply the same changes on the next sync. In the case of a push transaction, the changes are reapplied to the client on the next sync. If the client cannot push or pull the same changes, it must force a slow sync by sending [clientWantsToPushAllRecordsForEntityNames:](#) (page 68) to the session before the next sync.

Use this method at any time but the receiver should be released soon afterward because you cannot continue using a canceled session.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [isCancelled](#) (page 69)

- [finishSyncing](#) (page 69)

Related Sample Code

People

Declared In

ISyncSession.h

changeEnumeratorForEntityNames:

Returns the object enumerator for the ISyncChange objects which contain all the changes the client should apply to its local data.

- (NSEnumerator *)changeEnumeratorForEntityNames:(NSArray *)entityNames

Discussion

Use the returned object to iterate through the record changes during the pulling state. The *entityNames* parameter can contain a subset of the supported entities.

The sync engine applies client filters to the returned changes so that the client does not receive changes that were rejected. See *Sync Services Programming Guide* for more information on setting filters.

When the client applies a change described by an `ISyncChange` object, it must either accept or reject the change. You accept a change by sending `clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:` (page 62) to the sync session, and reject a change by sending `clientRefusedChangesForRecordWithIdentifier:` (page 67) to the sync session. After processing all changes (saving them on the device or locally) you send `clientCommittedAcceptedChanges` (page 64) to the sync session to commit those changes. Any uncommitted accepted changes or rejected changes are sent again to the client during the next sync.

Use this method during the pulling state only, otherwise an exception is raised.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `prepareToPullChangesForEntityNames:beforeDate:` (page 70)

Related Sample Code

People

SeeMyFriends

Declared In

`ISyncSession.h`

`clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:`

Informs the sync engine that the client has accepted the changes to the record identified by `recordIdentifier` during the pulling state.

```
- (void)clientAcceptedChangesForRecordWithIdentifier:(NSString *)recordIdentifier
    formattedRecord:(NSDictionary *)formattedRecord newRecordIdentifier:(NSString
    *)newRecordIdentifier
```

Discussion

If your client or device cannot store properties using the defined schema, then you can use the `formattedRecord` parameter to specify an alternate format. By specifying an alternate format, you assist the sync engine in figuring out which records and properties are equal during the mingling state, so that the sync engine doesn't generate false changes for records that were simply reformatted.

The `formattedRecord` dictionary should contain the entire record—key-value pairs for the formatted and unformatted properties that the client stores and pushes. This may be a subset of the entity properties if a client doesn't use the omitted properties. This dictionary should include the `ISyncRecordEntityNameKey` (page 75) key identifying the record's entity name. Otherwise, the `formattedRecord` parameter should be `nil`. For example, if you are syncing a device and the device truncates first and last names to 20 characters long, then you should specify a `formattedRecord` record containing the truncated values when invoking this method. See *Sync Services Programming Guide* for more details on formatting records.

The sync engine creates a new identifier for added records using `CFUUIDRef`. However, if your client generates its own unique record identifiers, then you can use the `newRecordIdentifier` parameter to request that your identifier be used in future communications with the sync engine. Although conflicts can occur when changing record identifiers that are targets of relationships pulled in the same sync session. Read *Syncing Relationships* in *Sync Services Programming Guide* for more information on resolving pulled relationships.

You can use this method to batch up changes by invoking it repeatedly. You can also use the `clientRefusedChangesForRecordWithIdentifier:` (page 67) method to reject changes. However, after a sequence of accepting and rejecting changes, you need to invoke `clientCommittedAcceptedChanges` (page 64) to end the transaction and actually commit them.

Use this method during the pulling state only, otherwise an exception is raised.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `clientChangedRecordIdentifiers:` (page 63)
- `clientLostRecordWithIdentifier:shouldReplaceOnNextSync:` (page 67)

Related Sample Code

People

SeeMyFriends

Declared In

`ISyncSession.h`

clientChangedRecordIdentifiers:

Changes the record identifiers of the given records.

```
- (void)clientChangedRecordIdentifiers:(NSDictionary *)oldToNew
```

Discussion

The `oldToNew` dictionary should contain key-value pairs where the keys are the old record identifiers, currently used by the sync engine, and the values are the new record identifiers. Use this method if your application generates its own unique record identifier. Alternatively, you can change individual identifiers when adding a record during the pulling state by sending `clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:` (page 62) to the session. This method can be invoked when pushing or pulling records.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

StickiesExample

Declared In

`ISyncSession.h`

clientCommittedAcceptedChanges

Informs the sync engine that all accepted and rejected changes in the current transaction during the pulling state should be committed.

- (void)clientCommittedAcceptedChanges

Discussion

Invoke this method after you save the accepted changes locally or on the device you are syncing.

You accept a change by sending

`clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:` (page 62) to the sync session, and reject a change by sending

`clientRefusedChangesForRecordWithIdentifier:` (page 67) to the sync session.

Invoking this method ends an open pull transaction that began by sending either

`prepareToPullChangesForEntityNames:beforeDate:` (page 70) or

`prepareToPullChangesInBackgroundForEntityNames:target:selector:` (page 71) to the receiver.

Once a transaction ends the sync engine commits the changes to the truth database, and opens a new pull transaction.

Use this method during the pulling state only, otherwise an exception is raised.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `cancelSyncing` (page 61)

- `finishSyncing` (page 69)

Related Sample Code

People

SeeMyFriends

Declared In

ISyncSession.h

clientCommittedAcceptedChangesWithNextAnchors:

Sets the sync anchors for each entity whose records were successfully updated during the pulling phase.

- (void)clientCommittedAcceptedChangesWithNextAnchors:(NSDictionary *)anchors

Parameters

anchors

The sync anchors for the entities that were successfully pulled from the sync engine.

The keys are the entity names and the values are the sync anchors. Sync anchors are globally unique `NSString` objects, typically containing a UUID or date. If this parameter is `nil`, the client refresh syncs.

As a convenience, you may use the same sync anchor for all entities of a data class by specifying a sync anchor for only one entity in that data class. If you provide sync anchors for two or more entities per data class, then you need to specify sync anchors for all entities in that data class. A missing sync anchor for an entity causes that entity to refresh sync.

Discussion

If you are using sync anchors, create a new sync anchor per entity or data class that is successfully pulled from the sync engine, store them locally, and invoke this method to register the new sync anchors with the sync engine. Then pass these new sync anchors to the sync engine at the beginning of the next sync by invoking either the [beginSessionWithClient:entityNames:beforeDate:lastAnchors:](#) (page 59) or [beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:](#) (page 56) methods. An exception is raised if this method is invoked on a session that was not created using one of these `...lastAnchors:` methods.

Use the [clientFinishedPushingChangesWithNextAnchors:](#) (page 66) method to change the sync anchors for entities that are successfully pushed.

Availability

Available in Mac OS X v10.5 and later.

See Also

- + [beginSessionWithClient:entityNames:beforeDate:lastAnchors:](#) (page 59)
- + [beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:](#) (page 56)
- [clientFinishedPushingChangesWithNextAnchors:](#) (page 66)

Declared In

`ISyncSession.h`

clientDidResetEntityNames:

Tells the sync engine to perform a refresh sync of all the records for the specified entities.

```
- (void)clientDidResetEntityNames:(NSArray *)entityNames
```

Discussion

The *entityNames* array can contain a subset of the supported entity names. Use this method if a user hard-resets a device by removing all its records, or if the local client data file is accidentally deleted.

After invoking this method, the client is expected to push all the records for the specified entities similar to a slow sync. However, during a refresh sync, the sync engine resets the client's sync state (as if it never synced before) and consequently, does not generate any delete changes when the client pulls records. Although the client may pull delete changes if the sync engine detects duplicate records.

Use this method during the negotiation state only, otherwise an exception is raised.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [shouldPullChangesForEntityName:](#) (page 73)
- [shouldReplaceAllRecordsOnClientForEntityName:](#) (page 74)

Related Sample Code

People

Declared In

`ISyncSession.h`

clientFinishedPushingChangesWithNextAnchors:

Sets the sync anchors for each entity whose records were successfully pushed by the client.

```
- (void)clientFinishedPushingChangesWithNextAnchors:(NSDictionary *)anchors
```

Parameters

anchors

The sync anchors for the entities that were successfully pushed to the sync engine.

The keys are the entity names and the values are the sync anchors. Sync anchors are globally unique `NSString` objects, typically containing a UUID or date. If this parameter is `nil`, the client refresh syncs.

As a convenience, you may use the same sync anchor for all entities of a data class by specifying a sync anchor for only one entity in that data class. If you provide sync anchors for two or more entities per data class, then you need to specify sync anchors for all entities in that data class. A missing sync anchor for an entity causes that entity to refresh sync.

Discussion

If you are using sync anchors, create a new sync anchor per entity or data class that is successfully pushed to the sync engine, store them locally, and invoke this method to register the new sync anchors with the sync engine. Then pass these new sync anchors to the sync engine at the beginning of the next sync by invoking either the `beginSessionWithClient:entityNames:beforeDate:lastAnchors:` (page 59) or `beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:` (page 56) methods. An exception is raised if this method is invoked on a session that was not created using one of these methods.

Use the `clientCommittedAcceptedChangesWithNextAnchors:` (page 64) method to change the sync anchors for entities that are successfully pulled.

Availability

Available in Mac OS X v10.5 and later.

See Also

+ `beginSessionWithClient:entityNames:beforeDate:lastAnchors:` (page 59)

+ `beginSessionInBackgroundWithClient:entityNames:target:selector:lastAnchors:` (page 56)

- `clientCommittedAcceptedChangesWithNextAnchors:` (page 64)

Declared In

`ISyncSession.h`

clientInfoForRecordWithIdentifier:

Returns a client-specific, nonprepsynchronized object that stores additional information about a record specified by *recordIdentifier*.

```
- (id)clientInfoForRecordWithIdentifier:(NSString *)recordIdentifier
```

Discussion

Returns `nil` if the record has no client information or doesn't exist.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [setClientInfo:forRecordWithIdentifier:](#) (page 73)

Declared In

ISyncSession.h

clientLostRecordWithIdentifier:shouldReplaceOnNextSync:

Tells the sync engine that a record identified by *recordIdentifier*, no longer exists on the client, and indicates whether or not it should be replaced.

```
- (void)clientLostRecordWithIdentifier:(NSString *)recordIdentifier
    shouldReplaceOnNextSync:(BOOL)flag
```

Discussion

If this method is invoked during the pushing state and *flag* is YES, then the record is added during the subsequent pulling state in the same sync session. However, if this method is invoked during the pulling state (pushing has already taken place), it is added the next time the client syncs.

If *flag* is NO, the sync engine treats the record as if it had been filtered and does not send any more changes for the record. If invoked during the pulling state, this is equivalent to refusing a record using the [clientRefusedChangesForRecordWithIdentifier:](#) (page 67) method.

Use this method if you inadvertently deleted a record when pushing or pulling changes. For example, use this method if the contacts on a phone device are full and the user must select a record to delete in order to add a record. The user's only choice is to delete an existing record on the phone but they don't want the record deleted from the Address Book on their computer.

You can use this method during the pulling and pushing state.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [deleteRecordWithIdentifier:](#) (page 68)

- [clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:](#) (page 62)

Declared In

ISyncSession.h

clientRefusedChangesForRecordWithIdentifier:

Informs the sync engine during the pulling state that the client has refused to apply the changes for the record specified by *recordIdentifier*.

```
- (void)clientRefusedChangesForRecordWithIdentifier:(NSString *)recordIdentifier
```

Discussion

This method applies only to add and modify changes, not deletes. After invoking this method, the sync engine does not send the same change during any subsequent syncs unless the record is modified. Refusing a record does not change the local identifier mapping for the client. Invoking this method does not affect other clients participating in the same sync session.

Note that if a client refresh syncs, the entire client store is wiped out, so any previously refused records are reapplied to the client. Use filtering if you want to permanently ignore some records. Do not use this method if you want to refuse deletes. Instead push the records that you want to keep during the next sync session.

Use this method during the pulling state only, otherwise an exception is raised.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:](#) (page 62)
- [clientCommittedAcceptedChanges](#) (page 64)
- [clientLostRecordWithIdentifier:shouldReplaceOnNextSync:](#) (page 67)

Declared In

ISyncSession.h

clientWantsToPushAllRecordsForEntityNames:

Forces a slow sync of all the records for the specified entities.

```
- (void)clientWantsToPushAllRecordsForEntityNames:(NSArray *)entityNames
```

Discussion

The `entityNames` array can contain a subset of the supported entity names. A **slow sync** is when the client pushes all of its records to the sync engine, and the sync engine determines, by comparing records, what changes need to be applied and pushed to the client. By default, the sync engine assumes the client is **fast syncing** and expects the client to push only the changes to records since the last sync (added, modified, and deleted records). Use this method if you want to change this default behavior by forcing a slow sync. For example, if the client can't determine what records changed.

Use this method during the negotiation state only, otherwise an exception is raised.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [shouldPushAllRecordsForEntityName:](#) (page 73)
- [shouldPushChangesForEntityName:](#) (page 74)

Related Sample Code

People

Declared In

ISyncSession.h

deleteRecordWithIdentifier:

Creates a delete change for the record specified by `recordIdentifier` and pushes the change to the sync engine.

- (void)deleteRecordWithIdentifier:(NSString *)*recordIdentifier*

Discussion

Use this method during the negotiation state or pushing state only, otherwise an exception is raised. Invoking this method in the negotiation state transitions the receiver to the pushing state.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [pushChange:](#) (page 71)
- [pushChangesFromRecord:withIdentifier:](#) (page 72)
- [clientLostRecordWithIdentifier:shouldReplaceOnNextSync:](#) (page 67)
- [shouldPushAllRecordsForEntityName:](#) (page 73)

Related Sample Code

People

Declared In

ISyncSession.h

finishSyncing

Tells the sync engine that the client is done syncing. Invoking this method closes any open transactions in the pushing or pulling states.

- (void)finishSyncing

Discussion

You must invoke this method to cleanly terminate the session.

Use this method at any time but the receiver should be released soon afterwards since you cannot continue using a finished session.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [cancelSyncing](#) (page 61)
- [isCancelled](#) (page 69)

Related Sample Code

People

SeeMyFriends

StickiesExample

Declared In

ISyncSession.h

isCancelled

Returns YES if the receiver was canceled, NO otherwise.

- (BOOL)isCancelled

Discussion

You cannot continue using a canceled session.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [cancelSyncing](#) (page 61)
- [finishSyncing](#) (page 69)

Declared In

ISyncSession.h

prepareToPullChangesForEntityNames:beforeDate:

Moves the receiver to the mingling state and returns when the sync engine is ready for the client to begin pulling changes to the specified entities.

```
- (BOOL)prepareToPullChangesForEntityNames:(NSArray *)entityNames beforeDate:(NSDate *)date
```

Parameters

entityNames

The entity names to use in the pulling phase. Can be a subset of the supported entity names.

date

The date/time that the client is willing to wait for the mingling process to complete.

Return Value

Returns NO if *date* expires before the sync engine is ready for the client to begin pulling changes, YES otherwise. If this method returns NO, the pushed changes and sync anchors are saved but not applied until the next time the client syncs. If this method returns YES, you can invoke it multiple times until it returns YES or the sync session is canceled or finished. If this method returns YES, the mingling state ends.

Discussion

Invoke this method after you have finished pushing changes to the sync engine and want to begin pulling changes.

This method raises an exception if validation errors occur when saving pushed changes or the sync session is canceled. Similar to this method returning NO, when an `ISyncSessionCancelledException` exception is raised, the pushed changes and anchors are saved but not applied until the next sync. If a validation exception is raised, the pushed changes and anchors are not saved. If you pass a zero delay, `[Date date]`, as the *date* parameter, you can use this method to validate your pushed changes.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [prepareToPullChangesInBackgroundForEntityNames:target:selector:](#) (page 71)

Related Sample Code

SeeMyFriends

Declared In

ISyncSession.h

prepareToPullChangesInBackgroundForEntityNames:target:selector:

Moves the receiver to the mingling state and sends a message to a specified target when the sync engine is ready for the client to begin pulling changes to the specified entities.

```
- (void)prepareToPullChangesInBackgroundForEntityNames:(NSArray *)entityNames
    target:(id)target selector:(SEL)selector
```

Parameters*entityNames*

The entity names to use in the pulling phase. Can be a subset of the supported entity names.

target

The object to send *selector* to when the mingling process is complete.

selector

The message to send to *target* when the mingling process is complete. The *selector* method is passed two parameters: The first parameter is the `ISyncClient` object and the second parameter is the `ISyncSession` object.

Discussion

Use this method during the pushing state to transition to the mingling state. The mingling state ends when *selector* is sent to *target*. Use the [cancelSyncing](#) (page 61) method to cancel this method and the entire sync session. If the session is canceled, *selector* is sent to *target* passing `nil` as the `ISyncSession` object. This method may raise an exception before returning if a validation error occurs.

Note: `ISyncSession` is not thread-safe. You can pass an `ISyncSession` object between threads but you should not use it concurrently. Asynchronous callbacks from `ISyncSession` are delivered to any client thread that used any of the Sync Services methods. The client is responsible for directing the callback to an appropriate thread.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [prepareToPullChangesForEntityNames:beforeDate:](#) (page 70)

Declared In

ISyncSession.h

pushChange:

Pushes changes made to a single record, specified by *change*, to the sync engine.

```
- (void)pushChange:(ISyncChange *)change
```

Discussion

A client can push only one `ISyncChange` object per record. The `change` object encapsulates an add, modify, or delete record change. If the change is an add or modify, the `change` object can contain changes to multiple properties including deleting properties. The change is not actually pushed to the sync engine until the sync session leaves the pushing state.

When slow syncing, a client should push add changes only. When fast syncing, a client should push only the delta changes since the last time the client synced. These changes may include new records, modified records, and deleted records.

You can also delete records without creating an `ISyncChange` object using the `deleteRecordWithIdentifier:` (page 68) method. Use the `pushChangesFromRecord:withIdentifier:` (page 72) method if your client knows a record changed but doesn't keep track of individual property changes.

Use this method during the negotiation or pushing state only, otherwise an exception is raised. Invoking this method during the negotiation state transitions to the pushing state.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `shouldPushAllRecordsForEntityName:` (page 73)
- `shouldPushChangesForEntityName:` (page 74)

Declared In

`ISyncSession.h`

pushChangesFromRecord:withIdentifier:

Compares `record` to the client's previous known state of the record, identified by `recordIdentifier`, and pushes the changes to the sync engine.

```
- (void)pushChangesFromRecord:(NSDictionary *)record withIdentifier:(NSString *)recordIdentifier
```

Discussion

Use this method if you know a record changed but don't know what properties changed. Otherwise, use the `pushChange:` (page 71) method to specify the exact property changes made to this record.

Use this method during the negotiation or pushing state only, otherwise an exception is raised. Invoking this method during the negotiation state transitions to the pushing state.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `shouldPushAllRecordsForEntityName:` (page 73)
- `shouldPushChangesForEntityName:` (page 74)

Related Sample Code

People

Declared In

ISyncSession.h

setClientInfo:forRecordWithIdentifier:

Associates a client-specific, non synchronized object, *clientInfo*, to a record specified by *recordIdentifier*.

```
- (void)setClientInfo:(id <NSCoding>)clientInfo forRecordWithIdentifier:(NSString *)recordIdentifier
```

Discussion

The *clientInfo* parameter can be any object that conforms to the NSCoding protocol, and are deleted when the record is deleted. Pass *nil* for *clientInfo* to remove a previously set client information object. Use this method to store additional information with a record.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [clientInfoForRecordWithIdentifier:](#) (page 66)

Declared In

ISyncSession.h

shouldPullChangesForEntityName:

Returns YES if the client should pull changes to records for *entityName*, NO otherwise.

```
- (BOOL)shouldPullChangesForEntityName:(NSString *)entityName
```

Discussion

However, a return value of YES, doesn't imply the sync engine has changes for the entity. The sync engine doesn't know if there are any changes until after the mingling state. Use this method to determine if the client should try to pull changes for *entityName*. You can invoke this method at any time.

Availability

Available in Mac OS X v10.4 and later.

See Also

- [shouldReplaceAllRecordsOnClientForEntityName:](#) (page 74)

- [clientDidResetEntityNames:](#) (page 65)

Related Sample Code

People

Declared In

ISyncSession.h


shouldPushAllRecordsForEntityName:

Returns YES if the client should push all the records for *entityName* to the sync engine; otherwise, NO.

- (BOOL)shouldPushAllRecordsForEntityName:(NSString *)*entityName*

Discussion

For example, returns YES if you previously sent `clientWantsToPushAllRecordsForEntityNames:` (page 68) or `clientDidResetEntityNames:` (page 65) to the session to force a slow sync. This method also returns YES if the sync engine decides to slow sync *entityName*. If this method returns NO, the client should only push changes made since the last sync. You can invoke this method at any time.

 **Warning:** If this method returns YES and the client does not push a record that the client was known to have on the last sync, the sync engine assumes the record was deleted and deletes it from the truth.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `shouldPushChangesForEntityName:` (page 74)
- `pushChange:` (page 71)
- `pushChangesFromRecord:withIdentifier:` (page 72)
- `deleteRecordWithIdentifier:` (page 68)

Declared In

ISyncSession.h

shouldPushChangesForEntityName:

Returns YES if the client should push changes to records for *entityName* since the last sync, NO otherwise.

- (BOOL)shouldPushChangesForEntityName:(NSString *)*entityName*

Discussion

For example, this method returns NO if you previously sent `setShouldReplaceClientRecords:forEntityNames:` (page 27) to the client for this session passing YES as the *flag* parameter. If this method returns NO, the sync engine does not accept any changes from the client for this entity. You can invoke this method at any time.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `shouldPushAllRecordsForEntityName:` (page 73)
- `clientWantsToPushAllRecordsForEntityNames:` (page 68)

Declared In

ISyncSession.h

shouldReplaceAllRecordsOnClientForEntityName:

Returns YES if the client should delete all the records for the entity, specified by *entityName*, and replace them with records pulled from the sync engine, NO otherwise.

- (BOOL)shouldReplaceAllRecordsOnClientForEntityName:(NSString *)*entityName*

Discussion

Send `setShouldReplaceClientRecords:forEntityNames:` (page 27) to the client for this session to request a different behavior.

You can invoke this method at any time. However, you should not delete the local client data until the session enters the pulling state (after `prepareToPullChangesForEntityNames:beforeDate:` (page 70) returns). Otherwise, if the session is canceled prematurely, the client may be left in a non-synchronized state with no data.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `shouldPullChangesForEntityName:` (page 73)
- `clientDidResetEntityNames:` (page 65)

Declared In

`ISyncSession.h`

snapshotOfRecordsInTruth

Returns an immutable snapshot of the records in the truth database.

- (`ISyncRecordSnapshot *`)`snapshotOfRecordsInTruth`

Discussion

Use this method if you are syncing and want a snapshot that is consistent with the sync session. Otherwise, you can create a snapshot at any time using the `snapshotOfRecordsInTruthWithEntityNames:usingIdentifiersForClient:` (page 40) `ISyncManager` method. Snapshots are useful if you want to compare records or perform queries.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`ISyncSession.h`

Constants

This constant is a key used by a record pushed to the sync engine.

Constant	Description
<code>ISyncRecordEntityNameKey</code>	Each record pushed to the sync engine must have a value for this key that specifies the record's entity name.

These are the exceptions that might be thrown by `ISyncSession` methods during a sync.

Constant	Description
<code>ISyncSessionCancelledException</code>	Thrown by any method if invoked after the session was canceled.
<code>ISyncSessionUnavailableException</code>	Thrown if a session cannot be created, for example, if a client is already syncing.
<code>ISyncInvalidEntityException</code>	Thrown if a client tries creating a session with an entity that does not exist.
<code>ISyncUnsupportedEntityException</code>	Thrown if a client tries creating a session with an entity that exists but the client does not support.
<code>ISyncInvalidRecordException</code>	Thrown if a client pushes a malformed record.

These keys are used in the `userInfo` dictionary when a `ISyncInvalidRecordException` (page 76) exception is raised.

Constant	Description
<code>ISyncInvalidRecordIdentifiersKey</code>	An array of the record identifiers that raised the exception.
<code>ISyncInvalidRecordReasonsKey</code>	A dictionary where keys are the invalid record identifies and the values are the reasons for the exception.
<code>ISyncInvalidRecordsKey</code>	A dictionary where the keys are the invalid record identifiers and the values are the property keys that raised the exception.

ISyncSessionDriver Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	ISyncSessionDriver.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Sync Services Programming Guide

Overview

An `ISyncSessionDriver` object encapsulates the complex process of syncing client records. Using `ISyncSessionDriver` is an alternative approach to creating and managing your own `ISyncClient` and `ISyncSession` objects. The driver takes care of the details by creating a client, registering schemas, and managing sync sessions. An `ISyncSessionDriver` object can be used for multiple sync operations.

An `ISyncSessionDriver` object uses an application-supplied data source object to provide application-specific information needed to manage a sync session. For example, during a sync session, a data source supplies records or changes to push and applies pulled changes to local records. Some data source methods are required and others are optional.

The driver also sends callback messages to a delegate before and after most phases of a sync session. A delegate may implement these callback methods to customize the behavior of sync sessions. For example, a delegate might verify changes, resolve relationships, and perform some local database operations. If no delegate is specified, the driver sends the delegate messages to the data source.

You create an `ISyncSessionDriver` object using the `sessionDriverWithDataSource:` (page 79) method, passing a data source as the argument. The `sessionDriverWithDataSource:` (page 79) method raises an exception if a data source does not implement required methods. Optionally, set the delegate to a different object using the `setDelegate:` (page 82) method. All delegate methods are optional.

You perform a sync operation by sending `sync` (page 84) or `startAsynchronousSync:` (page 83) to an `ISyncSessionDriver` object. These methods perform all the phases of a sync operation: negotiating, pushing, mingling, and pulling. You can access the `ISyncClient` and `ISyncSession` objects directly using the `client` (page 80) and `session` (page 82) methods. However, `session` (page 82) returns `nil` if there is no active sync session.

An `ISyncSessionDriver` object takes care of finishing and canceling a sync session. Therefore, you should not send `finishSyncing` (page 69) or `cancelSyncing` (page 61) directly to an `ISyncSession` object returned by the `session` (page 82) method. Instead, send `finishSyncing` (page 81) to an `ISyncSessionDriver` object to prematurely finish a sync session. If an error occurs during syncing, send `lastError` (page 81) to the driver to get an `NSError` object describing the error.

Tasks

Creating a Session Driver

- + `sessionDriverWithDataSource:` (page 79)
Creates and returns a new driver object with the specified data source object.

Syncing

- `sync` (page 84)
Syncs client records, specified by the data source, with the sync engine.
- `startAsynchronousSync:` (page 83)
Syncs client records, specified by the data source, in a separate thread.
- `finishSyncing` (page 81)
Notifies the sync engine that the client is done syncing.

Error Handling

- `lastError` (page 81)
Returns the error that occurred during the last sync session.

Getting and Setting Properties

- `dataSource` (page 80)
Returns the data source object for the receiver.
- `setDelegate:` (page 82)
Sets the receiver's delegate to the specified object.
- `delegate` (page 80)
Returns the receiver's delegate.
- `client` (page 80)
Returns the client object used by the receiver to perform the sync operation.
- `session` (page 82)
Returns the session object used to manage the sync session.
- `setHandlesSyncAlerts:` (page 83)
Specifies whether the receiver should handle sync alerts.

- [handlesSyncAlerts](#) (page 81)
Returns a Boolean value indicating whether the receiver handles sync alerts.

Controlling Sync Behavior

- [sessionDriver:didRegisterClientAndReturnError:](#) (page 85) *delegate method*
Informs the receiver that a client was registered.
- [sessionDriver:willPushAndReturnError:](#) (page 86) *delegate method*
Informs the receiver that client changes will be pushed to the sync engine.
- [sessionDriver:didPushAndReturnError:](#) (page 85) *delegate method*
Informs the receiver that client changes were pushed to the sync engine.
- [sessionDriver:willPullAndReturnError:](#) (page 86) *delegate method*
Informs the receiver that changes will be pulled from the sync engine.
- [sessionDriver:didPullAndReturnError:](#) (page 84) *delegate method*
Informs the receiver that changes were pulled from the sync engine.
- [sessionDriver:willFinishSessionAndReturnError:](#) (page 85) *delegate method*
Informs the receiver that a session will be finished.
- [sessionDriverDidFinishSession:](#) (page 86) *delegate method*
Informs the receiver that a session was finished.
- [sessionDriverWillCancelSession:](#) (page 87) *delegate method*
Informs the receiver that a session will be cancelled.
- [sessionDriverDidCancelSession:](#) (page 86) *delegate method*
Informs the receiver that a session was cancelled.

Class Methods

sessionDriverWithDataSource:

Creates and returns a new driver object with the specified data source object.

```
+ (ISyncSessionDriver *)sessionDriverWithDataSource:(id
<ISyncSessionDriverDataSource>)dataSource
```

Discussion

The *dataSource* argument must conform to the `ISyncSessionDriverDataSource` protocol. This method may raise an exception if required methods are not implemented. The [sync](#) (page 84) method sends messages to both the data source and delegate objects during a sync operation. If a delegate is not specified, then the data source also receives delegate messages.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [dataSource](#) (page 80)
- [sync](#) (page 84)

Declared In

ISyncSessionDriver.h

Instance Methods

client

Returns the client object used by the receiver to perform the sync operation.

- (ISyncClient *)client

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

dataSource

Returns the data source object for the receiver.

- (id <ISyncSessionDriverDataSource>)dataSource

Availability

Available in Mac OS X v10.5 and later.

See Also

+ [sessionDriverWithDataSource:](#) (page 79)

Declared In

ISyncSessionDriver.h

delegate

Returns the receiver's delegate.

- (id)delegate

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setDelegate:](#) (page 82)

Declared In

ISyncSessionDriver.h

finishSyncing

Notifies the sync engine that the client is done syncing.

- (void)finishSyncing

Discussion

Invoking this method closes any open transactions in the pushing or pulling states. You should use this method to prematurely finish a sync session. Do not send `finishSyncing` (page 69) directly to an `ISyncSession` object returned by the `session` (page 82) method.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `session` (page 82)
- `sync` (page 84)

Declared In

`ISyncSessionDriver.h`

handlesSyncAlerts

Returns a Boolean value indicating whether the receiver handles sync alerts.

- (BOOL)handlesSyncAlerts

Return Value

YES if the receiver handles sync alerts; otherwise, NO.

Discussion

By default, a session driver does not handle sync sessions. Use the `setHandlesSyncAlerts:` (page 83) method to turn this feature on or off.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `setHandlesSyncAlerts:` (page 83)

Declared In

`ISyncSessionDriver.h`

lastError

Returns the error that occurred during the last sync session.

- (NSError *)lastError

Discussion

Typically, you use this method to get the error if `sync` (page 84) returns NO or the sync session started by the `startAsynchronousSync:` (page 83) method fails. The value returned is only valid until the start of the next sync session. Get the last error as follows:

```
BOOL success = [sessionDriver sync];
if (success == NO) myError = [sessionDriver lastError];
```

Availability

Available in Mac OS X v10.5 and later.

See Also

- [startAsynchronousSync](#): (page 83)
- [sync](#) (page 84)

Declared In

ISyncSessionDriver.h

session

Returns the session object used to manage the sync session.

```
- (ISyncSession *)session
```

Discussion

Typically, you use this method to check whether a sync session is in progress. Session objects returned from this method are valid only during the invocation of the [sync](#) (page 84) method when a sync session is in progress. Otherwise, this method returns `nil`. If you retain a session object returned by this method, it is no longer valid after the `sync` method returns or after one of these delegate methods is invoked:

```
sessionDriverDidFinishSession:
sessionDriverDidCancelSession:
```

Use this method only during the same thread as the `sync` method.

You should not send [finishSyncing](#) (page 69) or [cancelSyncing](#) (page 61) directly to an `ISyncSession` object returned by this method. Send `finishSyncing` to an `ISyncSessionDriver` object to prematurely finish a sync session. Return an `NSError` object as one of the arguments to a delegate method to cancel a sync session.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [sync](#) (page 84)
- [finishSyncing](#) (page 81)

Declared In

ISyncSessionDriver.h

setDelegate:

Sets the receiver's delegate to the specified object.

```
- (void)setDelegate:(id)delegate
```

Discussion

The messages sent to a delegate are described in “Controlling Sync Behavior.” The delegate doesn’t need to implement all of these methods. If no delegate is set or the *delegate* argument is *nil*, delegate messages are sent to the data source object instead.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [delegate](#) (page 80)
- + [sessionDriverWithDataSource:](#) (page 79)

Declared In

`ISyncSessionDriver.h`

setHandlesSyncAlerts:

Specifies whether the receiver should handle sync alerts.

```
- (void)setHandlesSyncAlerts:(BOOL)flag
```

Parameters

flag

If YES, the receiver should handle sync alerts; otherwise, the receiver doesn’t handle sync alerts.

Discussion

A session driver may optionally handle sync alerts for a client. If the session driver handles sync alerts, then it registers a sync alert handler and receives notifications for requests to join sync sessions. When the session driver receives a request, it initiates a sync session as if the [startAsynchronousSync:](#) (page 83) method was invoked by the client so it doesn’t sync in the main thread. By default, a session driver does not handle sync sessions.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [startAsynchronousSync:](#) (page 83)
- [handlesSyncAlerts](#) (page 81)
- [sync](#) (page 84)

Declared In

`ISyncSessionDriver.h`

startAsynchronousSync:

Syncs client records, specified by the data source, in a separate thread.

```
- (BOOL)startAsynchronousSync:(NSError **)outError
```

Discussion

This method is similar to the [sync](#) (page 84) method but returns immediately while performing a sync session asynchronously. Use the delegate methods described in “Controlling Sync Behavior” (page 79) if you want to perform some operations at different phases during the sync session including receiving notification

when the sync session is finished or cancelled. If the driver is unable to create a sync session, this method returns `NO` and the `outError` argument is set to an `NSError` object describing the error; otherwise, this method returns `YES`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [sync](#) (page 84)
- [finishSyncing](#) (page 81)

Declared In

`ISyncSessionDriver.h`

sync

Syncs client records, specified by the data source, with the sync engine.

- `(BOOL)sync`

Discussion

This method registers a client, registers schemas, and manages an entire sync session. It begins a sync session, negotiates a sync mode, pushes records, pulls records, and ends the sync session. During a sync session the data source is expected to supply records or changes to push and to apply pulled changes to local records. Optionally, use the delegate methods described in “[Controlling Sync Behavior](#)” (page 79) if you want to perform some operations at different phases during the sync session. Use the [finishSyncing](#) (page 81) method to cancel a sync session started by this method. This method returns `YES` if the sync session is successful; otherwise, `NO`.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [startAsynchronousSync:](#) (page 83)
- [finishSyncing](#) (page 81)

Declared In

`ISyncSessionDriver.h`

Delegate Methods

sessionDriver:didPullAndReturnError:

Informs the receiver that changes were pulled from the sync engine.

- `(BOOL)sessionDriver:(ISyncSessionDriver *)sender didPullAndReturnError:(NSError **)outError`

Discussion

If an error occurs, this method returns `NO` and sets `outError` to an `NSError` object that describes the error.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

sessionDriver:didPushAndReturnError:

Informs the receiver that client changes were pushed to the sync engine.

```
- (BOOL)sessionDriver:(ISyncSessionDriver *)sender didPushAndReturnError:(NSError **)outError
```

Discussion

If an error occurs, this method returns *NO* and sets *outError* to an *NSError* object that describes the error.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

sessionDriver:didRegisterClientAndReturnError:

Informs the receiver that a client was registered.

```
- (BOOL)sessionDriver:(ISyncSessionDriver *)sender didRegisterClientAndReturnError:(NSError **)outError
```

Discussion

If an error occurs, this method returns *NO* and sets *outError* to an *NSError* object that describes the error.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

sessionDriver:willFinishSessionAndReturnError:

Informs the receiver that a session will be finished.

```
- (BOOL)sessionDriver:(ISyncSessionDriver *)sender willFinishSessionAndReturnError:(NSError **)outError
```

Discussion

If an error occurs, this method returns *NO* and sets *outError* to an *NSError* object that describes the error.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

sessionDriver:willPullAndReturnError:

Informs the receiver that changes will be pulled from the sync engine.

```
- (BOOL)sessionDriver:(ISyncSessionDriver *)sender willPullAndReturnError:(NSError **)outError
```

Discussion

If an error occurs, this method returns `NO` and sets `outError` to an `NSError` object that describes the error.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncSessionDriver.h`

sessionDriver:willPushAndReturnError:

Informs the receiver that client changes will be pushed to the sync engine.

```
- (BOOL)sessionDriver:(ISyncSessionDriver *)sender willPushAndReturnError:(NSError **)outError
```

Discussion

If an error occurs, this method returns `NO` and sets `outError` to an `NSError` object that describes the error.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncSessionDriver.h`

sessionDriverDidCancelSession:

Informs the receiver that a session was cancelled.

```
- (void)sessionDriverDidCancelSession:(ISyncSessionDriver *)sender
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncSessionDriver.h`

sessionDriverDidFinishSession:

Informs the receiver that a session was finished.

```
- (void)sessionDriverDidFinishSession:(ISyncSessionDriver *)sender
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

sessionDriverWillCancelSession:

Informs the receiver that a session will be cancelled.

- (void)sessionDriverWillCancelSession:(ISyncSessionDriver *)sender

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

NSPersistentStoreCoordinator Sync Services Additions Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncCoreData.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Sync Services Programming Guide

Overview

This category adds support for syncing Core Data applications—that is, syncing local changes made to managed objects to corresponding records managed by the sync engine—records that may be shared with other applications and synced to other computers over .Mac.

Use the [syncWithClientInBackground:handler:error:](#) (page 90) method to start a sync either in the foreground or background. Optionally provide a sync handler that can intervene during the sync session. For example, verify changes before they are applied to local managed objects. The supplied sync handler must conform to the `NSPersistentStoreCoordinatorSyncing` protocol.

Read Syncing Core Data Applications in *Sync Services Programming Guide* for more details on using Core Data sync.

Tasks

Syncing

- [syncWithClientInBackground:handler:error:](#) (page 90)
Syncs managed objects, stored by the receiver, with the sync engine.
- [setStoresFastSyncDetailsAtURL:forPersistentStore:](#) (page 90)
Specifies where to save details about fast syncing for a persistent store.

Instance Methods

setStoresFastSyncDetailsAtURL:forPersistentStore:

Specifies where to save details about fast syncing for a persistent store.

```
- (void)setStoresFastSyncDetailsAtURL:(NSURL *)url
    forPersistentStore:(NSPersistentStore *)store
```

Parameters

url

The location to store the information.

store

The persistent store that is syncing.

Discussion

In order to fast sync, the persistent store coordinator needs to store information about what entities to push in the next sync. Use this method to specify where, in the file system, to store information about a sync per persistent store. This method must be invoked before any of the persistent stores are changed. If one of the persistent stores are changed before this method is invoked, the persistent store coordinator slow syncs on the next sync.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncCoreData.h

syncWithClientInBackground:handler:error:

Syncs managed objects, stored by the receiver, with the sync engine.

```
- (BOOL)syncWithClient:(ISyncClient *)client inBackground:(BOOL)flag handler:(id
    < NSPersistentStoreCoordinatorSyncing >)syncHandler error:(NSError **)rError
```

Parameters

client

The client to sync.

flag

YES if syncing is non-blocking and occurs in a separate thread. NO if syncing blocks and occurs in the senders thread—in which case, this method returns when syncing finishes or cancels.

syncHandler

Application supplied object that can optionally intervene during the sync session.

outError

If the receiver fails to sync, an NSError object describing the error.

Return Value

NO if the receiver is unable to create a sync session or an error occurs during the sync; otherwise, YES.

Discussion

Pass YES for the `inBackground:` parameter if you do not want this method to block—for example, if you are handling a sync alert and you registered the sync alert handler on the main thread.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncCoreData.h`

Managers

Sync Services Constants Reference

Framework: SyncServices/SyncServices.h

Overview

This chapter describes the types and constants found in the Sync Services:

Constants

Enumerations

ISyncChange—Change Types

```
typedef int ISyncChangeType;
enum __ISyncChangeType {
    ISyncChangeTypeAdd=1,
    ISyncChangeTypeModify,
    ISyncChangeTypeDelete
};
```

Discussion

These types are described in [ISyncChange Class Reference](#) (page 11).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncChange.h

ISyncClient—Client Status

```
typedef SInt32 ISyncStatus;
enum __ISyncStatus {
    ISyncStatusRunning=1,
    ISyncStatusSuccess,
    ISyncStatusWarnings,
    ISyncStatusErrors,
    ISyncStatusCancelled,
    ISyncStatusFailed,
    ISyncStatusNever
};
```

Discussion

These types are described in [ISyncClient Class Reference](#) (page 17).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncClient.h

ISyncSessionDriver—Preferred Sync Mode

```
typedef enum {
    ISyncSessionDriverModeFast = 1,
    ISyncSessionDriverModeSlow,
    ISyncSessionDriverModeRefresh,
} ISyncSessionDriverMode;
```

Discussion

These types are described in [ISyncSessionDriverDataSource Protocol Reference](#) (page 107).

Availability

Available in Mac OS X v10.5 and later.

Declared In

SyncServices/ISyncSessionDriver.h

ISyncSessionDriver—Status of Applying Changes

```
typedef enum {
    ISyncSessionDriverChangeRefused = 0,
    ISyncSessionDriverChangeAccepted,
    ISyncSessionDriverChangeIgnored,
    ISyncSessionDriverChangeError
} ISyncSessionDriverChangeResult;
```

Discussion

These types are described in [ISyncSessionDriverDataSource Protocol Reference](#) (page 107).

Availability

Available in Mac OS X v10.5 and later.

Declared In

SyncServices/ISyncSessionDriver.h

Error Codes

Codes for errors that can occur when using an ISyncSessionDriver object or other Sync Services classes.

```
enum {
    ISyncSessionClientAlreadySyncingError =    100,
    ISyncSessionUserCanceledSessionError =    101,
    ISyncSessionDriverRegistrationError =      200,
    ISyncSessionDriverPullFailureError =      201,
    ISyncSessionDriverFatalError =            300
};
```

Constants

ISyncSessionClientAlreadySyncingError

Error code that indicates the client is already syncing.

Available in Mac OS X v10.5 and later.

Declared in SyncServicesErrors.h.

ISyncSessionUserCanceledSessionError

Error code that indicates the user canceled the sync session.

Available in Mac OS X v10.5 and later.

Declared in SyncServicesErrors.h.

ISyncSessionDriverRegistrationError

Error code that indicates the session driver failed to register the client.

Available in Mac OS X v10.5 and later.

Declared in SyncServicesErrors.h.

ISyncSessionDriverPullFailureError

Error code that indicates the session driver failed to pull records.

Available in Mac OS X v10.5 and later.

Declared in SyncServicesErrors.h.

ISyncSessionDriverFatalError

Error code that indicates an ISyncSessionDriver received a fatal error.

Available in Mac OS X v10.5 and later.

Declared in SyncServicesErrors.h.

Availability

Available in Mac OS X v10.5 and later.

Declared In

SyncServices/SyncServicesErrors.h

Global Variables

ISyncChange—Property Keys

```
extern NSString * const ISyncChangePropertyActionKey;  
extern NSString * const ISyncChangePropertySet;  
extern NSString * const ISyncChangePropertyClear;  
extern NSString * const ISyncChangePropertyNameKey;  
extern NSString * const ISyncChangePropertyValueKey;
```

Discussion

These constants are defined in [ISyncChange Class Reference](#) (page 11).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncChange.h

ISyncClient—Client Types

```
extern NSString * const ISyncClientTypeApplication;  
extern NSString * const ISyncClientTypeDevice;  
extern NSString * const ISyncClientTypeServer;  
extern NSString * const ISyncClientTypePeer;
```

Discussion

These constants are defined in [ISyncClient Class Reference](#) (page 17).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncClient.h

ISyncManager—Exceptions

```
extern NSString * const ISyncServerUnavailableException
```

Discussion

This constant is defined in [ISyncManager Class Reference](#) (page 35).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncManager.h

ISyncSession—Exceptions

```
extern NSString * const ISyncSessionCancelledException;
extern NSString * const ISyncSessionUnavailableException;
extern NSString * const ISyncInvalidRecordException;
extern NSString * const ISyncInvalidRecordIdentifiersKey;
extern NSString * const ISyncInvalidRecordReasonsKey;
extern NSString * const ISyncInvalidRecordsKey;
extern NSString * const ISyncInvalidEntityException;
extern NSString * const ISyncUnsupportedEntityException;
```

Discussion

These constants are defined in [ISyncSession Class Reference](#) (page 51).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncSession.h

ISyncSession—Key

```
extern NSString * const ISyncRecordEntityNameKey;
```

Discussion

These constants are defined in [ISyncSession Class Reference](#) (page 51).

Availability

Available in Mac OS X v10.4 and later.

Declared In

SyncServices/ISyncSession.h

ISyncErrorDomain

Domain for errors that occurred using Sync Services classes and methods.

```
NSString *const ISyncErrorDomain;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

SyncServices/SyncServicesErrors.h

Protocols

ISyncFiltering Protocol Reference

Conforms to	NSCoding
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncFilter.h
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Sync Services Programming Guide

Overview

ISyncFiltering is a protocol implemented by objects that filter records for a client. A client can filter the records it pulls from the sync engine using these objects. Before a record is pulled by a client, the sync engine passes it to each filter associated with the client. A filter can then accept or reject the record. If a filter rejects a record, it is not passed to the client.

For example, a user might want to sync only contacts with phone numbers to their mobile phone. The filter for the phone client would examine each contact and reject it if the contact has no phone number.

Use the [setFilters:](#) (page 25) ISyncClient method to set the filters for a client. The sync engine archives the filters so that they persist after the client process terminates. Because the filters persist, they must conform to the NSCoding protocol. In addition, any process that loads the filters using the [filters](#) (page 22) ISyncClient method must have the classes for those filters. When the set of filters changes, all records for a client must be re-filtered to determine if they need to be pulled during the next sync. This is a potentially computationally expensive operation, so only change filters when necessary.

Note

You can use the ISyncFilter class methods to combine multiple filters into a single filter using logical AND or OR binary operators on a set of filters.

Tasks

Testing for Equality

- [isEqual:](#) (page 104)

Returns YES if the receiver and *anotherFilter* are equal, NO otherwise.

Getting Supported Entities

- [supportedEntityNames](#) (page 105)
Returns an array of entity names that this filter supports.

Filtering Records

- [shouldApplyRecord:withRecordIdentifier:](#) (page 104)
Returns YES if the client should pull *record* uniquely identified by *recordIdentifier*, NO otherwise.

Instance Methods

isEqual:

Returns YES if the receiver and *anotherFilter* are equal, NO otherwise.

- (BOOL)isEqual:(id)anotherFilter

Discussion

When setting a filter using the [setFilters:](#) (page 25) ISyncClient method, the sync engine uses this method to compare the new filter with the previous filter (if there is one). If the filters are not equal, the sync engine recomputes all records that should be pushed to a client.

When setting filters, the sync engine compares the new filters with the old filters using `isEqual:`. If a filter has changed, the sync engine must refilter all the client's records—an expensive operation. Therefore, it's important that this implementation returns NO only if two filters differ in such a way that the records need to be refiltered.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncFilter.h

shouldApplyRecord:withRecordIdentifier:

Returns YES if the client should pull *record* uniquely identified by *recordIdentifier*, NO otherwise.

- (BOOL)shouldApplyRecord:(NSDictionary *)record withRecordIdentifier:(NSString *)recordIdentifier

Discussion

This is the method that implements the actual filtering logic.

Availability

Available in Mac OS X v10.4 and later.

Declared In

ISyncFilter.h

supportedEntityNames

Returns an array of entity names that this filter supports.

- (NSArray *)supportedEntityNames

Discussion

This filter is used only to filter records of the supported entities. An exception is raised if this method returns an empty array or `nil`.

Availability

Available in Mac OS X v10.4 and later.

Related Sample Code

People

Declared In

ISyncFilter.h

ISyncSessionDriverDataSource Protocol Reference

Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncSessionDriver.h
Availability	Available in Mac OS X v10.5 and later.

Overview

The `ISyncSessionDriverDataSource` protocol defines a set of methods that the data source of an `ISyncSessionDriver` object must implement. This document also includes optional informal protocol methods that a data source can implement.

A data source must implement the `clientIdentifier` (page 112) and `clientDescriptionURL` (page 112) methods in order for a driver to create an `ISyncClient` object. A data source must implement `preferredSyncModeForEntityName:` (page 115) to request a sync mode—for example, a slow sync when an application doesn't have information on changes since the last sync. However, as with any sync session, the sync engine decides on the actual sync mode used, which depends on many other factors.

A data source also needs to implement the `recordsForEntityName:moreComing:error:` (page 116) method to support slow syncing, and optionally, implement the `changedRecordsForEntityName:moreComing:error:` (page 110) or `changesForEntityName:moreComing:error:` (page 111) methods to support fast syncing. All of these pushing methods allow you to batch records and changes.

Similarly, the data source must implement the `applyChange:forEntityName:remappedRecordId:formattedRecord:error:` (page 109) and `deleteAllRecordsForEntityName:error:` (page 112) methods to apply changes during the pulling phase of a sync session.

Optional methods include `entityNamesToSync` (page 113) and `entityNamesToPull` (page 113) which can return a subset of the entities used by the client. The default is to push and pull records for all the entities provided in the client description.

You should use sync anchors to improve performance and avoid serious errors. A sync anchor is an object that is unique per client and per entity, that is saved periodically throughout a sync session. The sync engine compares the clients locally stored sync anchors with its copies to determine the next sync mode. For example, if there is a discrepancy in sync anchors, the client must slow sync. Read *ISyncSession Class Reference* for details.

To use sync anchors, implement the `lastAnchorForEntityName:` (page 114) method to return the previous sync anchor for the specified entity, and implement the `nextAnchorForEntityName:` (page 115) method to return the new sync anchor for the specified entity. It is your responsibility to save the sync anchors

returned by the `nextAnchorForEntityName:` (page 115) method locally, and return them in a subsequent call to the `lastAnchorForEntityName:` (page 114) method during the next sync session. Although these methods are optional, if you implement one, you must implement the other.

Tasks

Getting Client Information

- `clientIdentifier` (page 112)
Returns the client's unique identifier specified when registering the client.
- `clientDescriptionURL` (page 112)
Returns an NSURL object representing the path to the client description property list.
- `schemaBundleURLs` (page 117)
Returns an array containing NSURL objects representing the path to schemas this client uses.
- `entityNamesToSync` (page 113)
Returns an array of NSString objects representing the names of entities this client wants to sync.
- `entityNamesToPull` (page 113)
Returns an array of NSString objects representing the names of entities this client wants to pull.
- `sessionBeginTimeout` (page 117)
Returns the time, in seconds, that the client is willing to wait for a sync session to begin.
- `sessionPullChangesTimeout` (page 117)
Returns the time, in seconds, that the client is willing to wait for a sync session to mingle—that is, prepare to pull changes.

Negotiating

- `preferredSyncModeForEntityName:` (page 115)
Returns the client's preferred sync mode for the session.

Pulling

- `applyChange:forEntityName:remappedRecordIdentifier:formattedRecord:error:` (page 109)
Applies the given changes to a client's record during the pulling phase of a sync session.
- `deleteAllRecordsForEntityName:error:` (page 112)
Deletes all records for the specified entity.

Pushing

- `recordsForEntityName:moreComing:error:` (page 116)
Returns records for the given entity name that should be pushed to the sync engine during a slow sync.

- [changedRecordsForEntityName:moreComing:error:](#) (page 110)
Returns changed records for the given entity name that should be pushed to the sync engine during a fast sync.
- [changesForEntityName:moreComing:error:](#) (page 111)
Returns the changes to records that should be pushed to the sync engine during a fast sync.
- [identifiersForRecordsToDeleteForEntityName:moreComing:error:](#) (page 114)
Returns the record identifiers for deleted records that should be pushed to the sync engine during a fast sync.

Using Sync Anchors

- [lastAnchorForEntityName:](#) (page 114)
Returns the last sync anchor for the specified entity name.
- [nextAnchorForEntityName:](#) (page 115)
Returns the next sync anchor for the specified entity name.

Instance Methods

applyChange:forEntityName:remappedRecordIdentifier:formattedRecord:error:

Applies the given changes to a client's record during the pulling phase of a sync session.

```
(ISyncSessionDriverChangeResult)applyChange:(ISyncChange *)change
forEntityName:(NSString *)entityName remappedRecordIdentifier:(NSString
**)outRecordIdentifier formattedRecord:(NSDictionary **)outRecord error:(NSError
**)outError
```

Discussion

This method applies the changes from the truth database to the local copy of the record. The *change* parameter is an *ISyncChange* object that describes the changes to a record since the last sync.

If the change is of type *ISyncChangeTypeDelete* (page 15), then the *outRecordIdentifier* and *outRecord* parameters are ignored. Otherwise, they may be used to pass back additional information to the driver.

If the change is of type *ISyncChangeTypeAdd* (page 15) or *ISyncChangeTypeModify* (page 15) and this method accepts the change, then it may set the value referenced by *outRecordIdentifier* to an alternate local record identifier. The sync engine uses the returned local record identifier when communicating future changes.

If the change is of type *ISyncChangeTypeAdd* (page 15) or *ISyncChangeTypeModify* (page 15) and this method accepts the change, then it may specify an alternate format by setting the value referenced by *outRecord* to the new format. See [clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:](#) (page 62) for more details on formatting records.

This method returns [ISyncSessionDriverChangeAccepted](#) (page 118) if the change is accepted (successfully applied), [ISyncSessionDriverChangeRefused](#) (page 118) if it is refused, and [ISyncSessionDriverChangeIgnored](#) (page 118) if it is neither accepted nor refused. If a client refuses a change, the sync engine does not send the same change during any subsequent syncs unless the record is modified.

If an error occurs, this method returns [ISyncSessionDriverChangeError](#) (page 118) and sets *outError* to an NSError object that describes the error. If this method returns [ISyncSessionDriverChangeError](#) (page 118), the `ISyncSessionDriver` object that invoked this method cancels the sync session.

This method is invoked by the driver during the pulling phase of a sync session, after pushing records.

This method is required.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [clientAcceptedChangesForRecordWithIdentifier:formattedRecord:newRecordIdentifier:](#) (page 62) (`ISyncSession`)
- [clientRefusedChangesForRecordWithIdentifier:](#) (page 67) (`ISyncSession`)

Declared In

`ISyncSessionDriver.h`

changedRecordsForEntityName:moreComing:error:

Returns changed records for the given entity name that should be pushed to the sync engine during a fast sync.

```
- (NSDictionary *)changedRecordsForEntityName:(NSString *)entityName moreComing:(BOOL *)moreComing error:(NSError **)outError
```

Discussion

Returns a dictionary where the keys are record identifiers and the values are dictionary records that you want to push during a fast sync. The dictionary records must be suitable for pushing to the sync engine and belong to the entity specified by *entityName*. A dictionary record contains the properties of a record that you want to sync. All dictionary records must contain a value for the [ISyncRecordEntityNameKey](#) (page 75) key that identifies the record's entity. This method returns an empty dictionary if there are no records to push for this entity.

The *moreComing* parameter is used to batch records. If this method sets the value referenced by *moreComing* to YES, then this method is invoked repeatedly during the pushing phase of a sync session until *moreComing* is set to NO.

Because this method is invoked during a fast sync, it returns only records that changed since the last sync session. If this method batches the records, all changes should be returned by multiple invocations of this method before *moreComing* is set to NO. The sync engine compares each dictionary record with the previous version to determine which properties changed. Use the [changesForEntityName:moreComing:error:](#) (page 111) method instead if you know which properties changed.

If an error occurs, this method returns `nil` and sets `outError` to an `NSError` object that describes the error. If this method returns `nil`, the `ISyncSessionDriver` object that invoked this method cancels the sync session.

This method is invoked by the driver during the pushing phase of a sync session, before pulling records. A data source of an `ISyncSessionDriver` object is required to implement this method or the `changesForEntityName:moreComing:error:` (page 111) method if it requests a fast sync—that is, if the `preferredSyncModeForEntityName:` (page 115) method may return `ISyncSessionDriverModeFast` (page 118).

This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `changesForEntityName:moreComing:error:` (page 111)
- `preferredSyncModeForEntityName:` (page 115)

changesForEntityName:moreComing:error:

Returns the changes to records that should be pushed to the sync engine during a fast sync.

```
- (NSArray *)changesForEntityName:(NSString *)entityName moreComing:(BOOL *)moreComing error:(NSError **)outError
```

Discussion

Returns an array of `ISyncChange` objects describing the changes made to records since the last sync session. The array should only contain changes to records belonging to the entity specified by `entityName`. Returns an empty array if there are no changes to push for this entity.

The `moreComing` parameter is used to batch changes. If this method sets the value referenced by `moreComing` to YES, then this method is invoked repeatedly during the pushing phase of a sync session until `moreComing` is set to NO.

Because this method is invoked during a fast sync, it should return all changes since the last sync session. If this method batches the records, all changed records should be returned by multiple invocations of this method before `moreComing` is set to NO. Use the `changedRecordsForEntityName:moreComing:error:` (page 110) method instead if you don't know which properties changed.

If an error occurs, this method returns `nil` and sets `outError` to an `NSError` object that describes the error. If this method returns `nil`, the `ISyncSessionDriver` object that invoked this method cancels the sync session.

This method is invoked by the driver during the pushing phase of a sync session, before pulling records. A data source of an `ISyncSessionDriver` object is required to implement this method or the `changedRecordsForEntityName:moreComing:error:` (page 110) method if it requests a fast sync—that is, if the `preferredSyncModeForEntityName:` (page 115) method may return `ISyncSessionDriverModeFast` (page 118).

This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [changedRecordsForEntityName:moreComing:error:](#) (page 110)
- [preferredSyncModeForEntityName:](#) (page 115)

clientDescriptionURL

Returns an NSURL object representing the path to the client description property list.

- (NSURL *)clientDescriptionURL

Discussion

The client description property list specifies client information that the sync engine needs to know to sync its records. See *Sync Services Programming Guide* for a complete description of the client description file.

This method is required.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) (ISyncManager)

Declared In

ISyncSessionDriver.h

clientIdentifier

Returns the client's unique identifier specified when registering the client.

- (NSString *)clientIdentifier

Discussion

There are no restrictions on the content or length of the client identifier, but it must be unique across all clients. Typically, it's a DNS-style name such as `com.apple.iCal`.

This method is required.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [registerClientWithIdentifier:descriptionFilePath:](#) (page 38) (ISyncManager)

Declared In

ISyncSessionDriver.h

deleteAllRecordsForEntityName:error:

Deletes all records for the specified entity.


```
- (BOOL)deleteAllRecordsForEntityName:(NSString *)entityName error:(NSError
    **)outError
```

Discussion

Returns YES if the request to delete all records belonging to the entity specified by *entityName* is accepted, otherwise NO. This method only returns NO and sets *outError* to an NSError object that describes the error if a serious error occurred deleting all records.

This method is required.

Availability

Available in Mac OS X v10.5 and later.

Declared In

ISyncSessionDriver.h

entityNamesToPull

Returns an array of NSString objects representing the names of entities this client wants to pull.

```
- (NSArray *)entityNamesToPull
```

Discussion

Optionally implement this method to return the names of the entities to pull that must be a subset of the entity names returned by the [entityNamesToSync](#) (page 113) method. If this method is not implemented, the sync session pulls the entities returned by the [entityNamesToSync](#) (page 113) method. Returns an empty array if this client doesn't want to pull any entities.

This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [entityNamesToSync](#) (page 113)

entityNamesToSync

Returns an array of NSString objects representing the names of entities this client wants to sync.

```
- (NSArray *)entityNamesToSync
```

Discussion

Returns an empty array if this client doesn't want to sync any entities. The sync session pushes and pulls the entities returned by this method unless an alternate set of entities is specified by the optional [entityNamesToPull](#) (page 113) method. If this method is not implemented, the driver syncs all enabled entities contained in the client description property list.

This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [entityNamesToPull](#) (page 113)

identifiersForRecordsToDeleteForEntityName:moreComing:error:

Returns the record identifiers for deleted records that should be pushed to the sync engine during a fast sync.

```
- (NSArray *)identifiersForRecordsToDeleteForEntityName:(NSString *)entityName
    moreComing:(BOOL *)moreComing error:(NSError **)outError
```

Discussion

Returns an array of NSString objects representing the record identifiers of records that the client deleted since the last sync. The array should contain only identifiers of deleted records belonging to the entity specified by *entityName*. This method is invoked during a fast sync only. Returns an empty array if there are no deleted records to push for this entity.

The *moreComing* parameter is used to batch changes. If this method sets the value referenced by *moreComing* to YES, then this method is invoked repeatedly during the pushing phase of a sync session until *moreComing* is set to NO.

Alternatively, you can implement the [changedRecordsForEntityName:moreComing:error:](#) (page 110) method using the [ISyncChangeTypeDelete](#) (page 15) constant to denote a deleted record.

If an error occurs, this method returns *nil* and sets *outError* to an NSError object that describes the error. If this method returns *nil*, the [ISyncSessionDriver](#) object that invoked this method cancels the sync session.

This method is invoked by the driver during the pushing phase of a sync session, before pulling records. A data source of an [ISyncSessionDriver](#) object can optionally implement this method or the [changesForEntityName:moreComing:error:](#) (page 111) method using the [ISyncChangeTypeDelete](#) (page 15) constant to denote a deleted record.

This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [changesForEntityName:moreComing:error:](#) (page 111)

- [preferredSyncModeForEntityName:](#) (page 115)

lastAnchorForEntityName:

Returns the last sync anchor for the specified entity name.

```
- (NSString *)lastAnchorForEntityName:(NSString *)entityName
```

Parameters

entityName

An entity name.

Return Value

A sync anchor corresponding to the entity name that was saved locally and returned by the [nextAnchorForEntityName:](#) (page 115) method in the previous sync session.

Discussion

This method is invoked immediately after a sync session is created and before pushing records. This method is optional. However, if you implement this method you must also implement the [nextAnchorForEntityName:](#) (page 115) method.

See Also

- [nextAnchorForEntityName:](#) (page 115)

nextAnchorForEntityName:

Returns the next sync anchor for the specified entity name.

```
- (NSString *)nextAnchorForEntityName:(NSString *)entityName
```

Parameters

entityName

An entity name.

Return Value

A new sync anchor corresponding to the entity name that is saved locally.

Sync anchors must be globally unique `NSString` objects. Typically, sync anchors contain a UUID or date.

Discussion

This method is invoked once per entity name after pushing records just before mingling, and once per entity name after pulling records just before accepting changes. This method is optional. However, if you implement this method you must also implement the [lastAnchorForEntityName:](#) (page 114) method.

See Also

- [lastAnchorForEntityName:](#) (page 114)

preferredSyncModeForEntityName:

Returns the client's preferred sync mode for the session.

```
- (ISyncSessionDriverMode)preferredSyncModeForEntityName:(NSString *)entity
```

Discussion

Returns one of these constants that specifies the preferred sync mode for this client:

[ISyncSessionDriverModeFast](#) (page 118), [ISyncSessionDriverModeSlow](#) (page 118), and [ISyncSessionDriverModeRefresh](#) (page 118). This method is invoked by the driver during the negotiation phase of a sync session, before pushing records.

This method is required.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [clientDidResetEntityNames:](#) (page 65) (`ISyncSession`)

- `clientWantsToPushAllRecordsForEntityNames:` (page 68) (`ISyncSession`)
- `shouldPushChangesForEntityName:` (page 74) (`ISyncSession`)
- `shouldPushAllRecordsForEntityName:` (page 73) (`ISyncSession`)
- `shouldPullChangesForEntityName:` (page 73) (`ISyncSession`)
- `shouldReplaceAllRecordsOnClientForEntityName:` (page 74) (`ISyncSession`)

Declared In

`ISyncSessionDriver.h`

recordsForEntityName:moreComing:error:

Returns records for the given entity name that should be pushed to the sync engine during a slow sync.

```
- (NSDictionary *)recordsForEntityName:(NSString *)entityName moreComing:(BOOL *)moreComing error:(NSError **)outError
```

Discussion

Returns a dictionary where the keys are record identifiers and the values are dictionary records that you want to push. The dictionary records must be suitable for pushing to the sync engine and belong to the entity specified by `entityName`. A dictionary record contains the properties of a record that you want to sync. All dictionary records must contain a value for the `ISyncRecordEntityNameKey` (page 75) key that identifies the record's entity. This method returns an empty dictionary if there are no records to push for this entity.

The `moreComing` parameter is used to batch records. If this method sets the value referenced by `moreComing` to YES, then this method is invoked repeatedly during the pushing phase of a sync session until `moreComing` is set to NO.

Because this method is invoked during a slow sync, it should return all records for the given entity. If this method batches the records, all records should be returned by multiple invocations of this method before `moreComing` is set to NO. Otherwise, the sync engine assumes you deleted the records that you did not push and data loss may result.

If an error occurs, this method returns `nil` and sets `outError` to an `NSError` object that describes the error. If this method returns `nil`, the `ISyncSessionDriver` object that invoked this method cancels the sync session.

This method is invoked by the driver during the pushing phase of a sync session, before pulling records.

This method is required.



Warning: If this method does not return all the records that the client was known to have on the last sync, the sync engine assumes the record was deleted and deletes it from the truth database.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncSessionDriver.h`

schemaBundleURLs

Returns an array containing NSURL objects representing the path to schemas this client uses.

- (NSArray *)schemaBundleURLs

Discussion

A schema can define new entities and properties, and extend existing entities. A schema bundle may contain other files, such as images and localization files. The returned array should contain URLs for all the schemas—including the public schemas—that this client intends to use. See *Sync Services Programming Guide* for more details on the schema format and contents of a schema bundle.

This method is required.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [registerSchemaWithBundlePath:](#) (page 39) (ISyncManager)

Declared In

ISyncSessionDriver.h

sessionBeginTimeout

Returns the time, in seconds, that the client is willing to wait for a sync session to begin.

- (NSTimeInterval)sessionBeginTimeout

Discussion

The default value is 60.0 seconds. This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

+ [beginSessionWithClient:entityNames:beforeDate:](#) (page 58) (ISyncSession)

sessionPullChangesTimeout

Returns the time, in seconds, that the client is willing to wait for a sync session to mingle—that is, prepare to pull changes.

- (NSTimeInterval)sessionPullChangesTimeout

Discussion

The default value is 600.0 seconds. This method is optional.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [prepareToPullChangesForEntityNames:beforeDate:](#) (page 70) (ISyncSession)

Constants

Use these constants as possible return values for the [preferredSyncModeForEntityName:](#) (page 115) method. Read *Managing Your Sync Session* in *Sync Services Programming Guide* for a description of the different sync modes.

Constant	Description
<code>ISyncSessionDriver-ModeFast</code>	Indicates that the client wants to fast sync. Available in Mac OS X v10.5 and later. Declared in <code>ISyncSessionDriver.h</code> .
<code>ISyncSessionDriver-ModeSlow</code>	Indicates that the client wants to slow sync. If the client slow syncs, it needs to push every record. Available in Mac OS X v10.5 and later. Declared in <code>ISyncSessionDriver.h</code> .
<code>ISyncSessionDriver-ModeRefresh</code>	Indicates that the client wants to refresh sync. If the preferredSyncModeForEntityName: (page 115) method returns this constant, the <code>ISyncSessionDriver</code> object sends clientDidResetEntityNames: (page 65) to the <code>ISyncSession</code> object. Available in Mac OS X v10.5 and later. Declared in <code>ISyncSessionDriver.h</code> .

Use these constants as possible return values for the [applyChange:forEntityName:remappedRecordId:formattedRecord:error:](#) (page 109) method.

Constant	Description
<code>ISyncSessionDriverChangeRefused</code>	Indicates the client refused the change. Available in Mac OS X v10.5 and later. Declared in <code>ISyncSessionDriver.h</code> .
<code>ISyncSessionDriverChangeAccepted</code>	Indicates the client accepted the change. Available in Mac OS X v10.5 and later. Declared in <code>ISyncSessionDriver.h</code> .
<code>ISyncSessionDriverChangeIgnored</code>	Indicates the client ignored the change. Available in Mac OS X v10.5 and later. Declared in <code>ISyncSessionDriver.h</code> .
<code>ISyncSessionDriverChangeError</code>	Indicates an error occurred while applying the change. Declared in <code>ISyncSessionDriver.h</code> . Available in Mac OS X v10.5 and later.

ISyncUIHelper Protocol Reference

Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncUIHelper.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Sync Services Programming Guide

Overview

`ISyncUIHelper` is an informal protocol that an object implements to provide presentation details for custom schemas to the iSync user interface. If you created a custom sync schema and want more control over the display of your records in the iSync user interface—specifically, the data change alert and conflict resolver user interfaces—then add a class that implements this protocol to your schema bundle.

You specify the class that conforms to this protocol in the sync schema using the `UIHelperClass` key. When the sync schema bundle is loaded, an instance of the `UIHelperClass` class is created to handle the presentation requests. The `UIHelperClass` class must inherit from `NSObject` or one of its subclasses. The methods in this protocol are optional.

Implement the `attributedStringForPropertiesWithNames:... method to change the presentation of individual records. Implement the attributedStringForIdentityPropertiesWithNames:... method to change the presentation of a custom schema above the record in the user interface. For example, return an NSAttributedString object with a picture using NSTextAttachment or color the text.`

Read *Creating a Sync Schema* in *Sync Services Programming Guide* for a complete description of a sync schema.

Tasks

Customizing the Presentation of Schemas and Records

- `attributedStringForPropertiesWithNames:inRecord:comparisonRecords:defaultAttributes:` (page 121)
Provides a custom presentation of a record to the data change alert and conflict resolver user interfaces.
- `attributedStringForIdentityPropertiesWithNames:inRecord:comparisonRecords:firstLineAttributes:secondLineAttributes:` (page 120)
Provides a custom presentation of a schema to the data change alert and conflict resolver user interfaces.

Instance Methods

attributedStringForIdentityPropertiesWithNames:inRecord:comparisonRecords: firstLineAttributes:secondLineAttributes:

Provides a custom presentation of a schema to the data change alert and conflict resolver user interfaces.

```
- (NSAttributedString *)attributedStringForIdentityPropertiesWithNames:(NSArray
    *)propertyNames inRecord:(NSDictionary *)record comparisonRecords:(NSArray
    *)comparisonRecords firstLineAttributes:(NSDictionary *)firstLineAttributes
    secondLineAttributes:(NSDictionary *)secondLineAttributes
```

Parameters

propertyNames

The names of the identity properties of *record*.

record

The record to be displayed.

comparisonRecords

An array of record dictionaries. The records are those displayed alongside this record in the user interface. If the data change alert invokes this method, this array always contains 0 or 1 records—0 if the type of change is an add or delete and 1 if the type of change is a modify. If the conflict resolver invokes this method, the array contains $n-1$ records where n is the number of clients involved in the conflict including this client.

firstLineAttributes

The recommended `NSAttributedString` attributes to apply to the first line of the returned attributed string.

secondLineAttributes

The recommended `NSAttributedString` attributes to apply to the second line of the returned attributed string.

Return Value

An `NSAttributedString` object for display of a schema in the data change alert or conflict resolver user interface.

The object returned should contain two lines. The first line should have the recommended attributes specified by *firstLineAttributes*, and the second line should have the attributes specified by *secondLineAttributes*.

If this method returns a value other than `nil` when multiple names are in *propertyNames*, the returned string is used to represent all the identity properties. Only one value is displayed for all the properties, next to the localized display name of the first property.

If this method returns `nil` when *propertyNames* contains more than one property, this method is invoked once for each identity property in *propertyNames*.

Availability

Available in Mac OS X v10.5 and later.

attributedStringForPropertiesWithNames:inRecord:comparisonRecords:defaultAttributes:

Provides a custom presentation of a record to the data change alert and conflict resolver user interfaces.

```
- (NSAttributedString *)attributedStringForPropertiesWithNames:(NSArray
*)propertyNames inRecord:(NSDictionary *)record comparisonRecords:(NSArray
*)comparisonRecords defaultAttributes:(NSDictionary *)defaultAttributes
```

Parameters

propertyNames

The names of the properties of *record* to be displayed.

This array contains only one property name unless the property has dependent properties as defined in the schema. If it has dependent properties, the property along with its dependent property names is in this array.

record

The record to be displayed.

comparisonRecords

An array of record dictionaries. The records are those displayed along side this record in the user interface. If the data change alert invokes this method, this array always contains 0 or 1 records—0 if the type of change is an add or delete and 1 if the type of change is a modify. If the conflict resolver invokes this method, the array contains $n-1$ records where n is the number of clients involved in the conflict including this client.

defaultAttributes

The recommended `NSAttributedString` attributes that should be applied to the returned `NSAttributedString` object.

Return Value

An `NSAttributedString` object for display of a record in the data change alert or conflict resolver user interface.

If the length of the `NSAttributedString` object is greater than 0, the string is displayed as is. If the length of the `NSAttributedString` object equals 0, the standard localized string for properties with no value is displayed. If this method returns `nil`, the default string is used.

If this method returns a value other than `nil` when multiple names are in *propertyNames*, the returned string is used to represent all the dependent properties. Only one value is displayed for all the properties, next to the localized display name of the first property.

If this method returns `nil` when *propertyNames* contains more than one property, this method is invoked once for each dependent property in *propertyNames*.

Availability

Available in Mac OS X v10.5 and later.

NSPersistentStoreCoordinatorSyncing Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/SyncServices.framework
Declared in	SyncServices/ISyncCoreData.h
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Sync Services Programming Guide

Overview

The `NSPersistentStoreCoordinatorSyncing` protocol defines callback messages that are sent to a sync handler while a Core Data application syncs. You set a sync handler when you start a sync session using the `syncWithClientInBackground:handler:error:` (page 90) `NSPersistentStoreCoordinator` method. The callback messages defined in this protocol are sent before and after most phases of a sync session. A sync handler may implement these optional methods to customize the behavior of sync sessions. For example, a sync handler might change records before their pushed to the sync engine or verify changes pulled from the sync engine before they are applied to managed objects.

Read Syncing Core Data Applications in *Sync Services Programming Guide* for more details on using Core Data sync.

Tasks

Getting Managed Contexts

- `managedObjectContextsToMonitorWhenSyncingPersistentStoreCoordinator:` (page 124)
Returns the managed object contexts that the receiver wants to monitor during the next sync session.
- `managedObjectContextsToReloadAfterSyncingPersistentStoreCoordinator:` (page 125)
Returns the managed object contexts that should be reloaded after the persistent store coordinator syncs.

Controlling Sync Behavior

- `persistentStoreCoordinatorShouldStartSyncing:` (page 131)
Returns whether or not the persistent store coordinator should start syncing.

- [persistentStoreCoordinator:willPushChangesInSyncSession:](#) (page 130)
Informs the receiver that client changes will be pushed to the sync engine.
- [persistentStoreCoordinator:didPushChangesInSyncSession:](#) (page 128)
Informs the receiver that client changes were pushed to the sync engine.
- [persistentStoreCoordinator:willPullChangesInSyncSession:](#) (page 129)
Informs the receiver that changes will be pulled from the sync engine.
- [persistentStoreCoordinator:didPullChangesInSyncSession:](#) (page 127)
Informs the receiver that changes were pulled from the sync engine.
- [persistentStoreCoordinator:didFinishSyncSession:](#) (page 127)
Informs the receiver that a session was finished.
- [persistentStoreCoordinator:didCancelSyncSession:error:](#) (page 126)
Informs the receiver that a session was cancelled.
- [persistentStoreCoordinator:willPushRecord:forManagedObject:inSyncSession:](#) (page 130)
Informs the receiver that client changes to a specific record will be pushed to the sync engine.
- [persistentStoreCoordinator:willDeleteRecordWithIdentifier:inSyncSession:](#) (page 129)
Informs the receiver that a specific record will be deleted during the pushing phase of a sync session.
- [persistentStoreCoordinator:willApplyChange:toManagedObject:inSyncSession:](#) (page 128)
Informs the receiver that pulled changes will be applied to a specific record during a sync session.
- [persistentStoreCoordinator:didApplyChange:toManagedObject:inSyncSession:](#) (page 125)
Informs the receiver that pulled changes were applied to a specific record during a sync session.
- [persistentStoreCoordinator:didCommitChanges:inSyncSession:](#) (page 126)
Informs the receiver that all applied changes were committed during a sync session.

Instance Methods

managedObjectContextsToMonitorWhenSyncingPersistentStoreCoordinator:

Returns the managed object contexts that the receiver wants to monitor during the next sync session.

- (NSArray
*)managedObjectContextsToMonitorWhenSyncingPersistentStoreCoordinator:(NSPersistentStoreCoordinator
*)*coordinator*

Parameters

coordinator

The persistent store coordinator being synced.

Return Value

An array containing the managed object contexts to monitor.

Discussion

The sync session uses this method to determine if pulled changes should be applied. Pulled changes are ignored—not applied to a record—if any of the managed contexts, returned by this method, changed the same record. In this case, the local changes are pushed in the next sync session and the sync engine is responsible for resolving any conflicts.

Conflicts can result if clients are allowed to change managed objects during a sync session or editing is not disabled during syncing. For example, conflicts can result if the user changes a managed object while the sync session is pulling changes to the same managed object. However, implementing this method does not handle all types of conflicts. The user may still modify a managed object after a sync session applies changes and before a sync session finishes. To avoid this, you should either not allow editing during a sync session, or be prepared to merge local changes with pulled changes after a sync session.

Therefore, although this method is optional, not implementing this method increases the risk of conflicts unless you sync synchronously or disable editing when syncing.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [managedObjectContextsToReloadAfterSyncingPersistentStoreCoordinator](#): (page 125)

Declared In

ISyncCoreData.h

managedObjectContextsToReloadAfterSyncingPersistentStoreCoordinator:

Returns the managed object contexts that should be reloaded after the persistent store coordinator syncs.

```
- (NSArray
    *)managedObjectContextsToReloadAfterSyncingPersistentStoreCoordinator:(NSPersistentStoreCoordinator
    *)coordinator
```

Parameters

coordinator

The persistent store coordinator being synced.

Return Value

An array containing the managed object contexts to reload.

Discussion

If you do not implement this method, it is your responsibility to reload managed object contexts after a sync.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [managedObjectContextsToMonitorWhenSyncingPersistentStoreCoordinator](#): (page 124)

Declared In

ISyncCoreData.h

persistentStoreCoordinator:didApplyChange:toManagedObject:inSyncSession:

Informs the receiver that pulled changes were applied to a specific record during a sync session.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    didApplyChange:(ISyncChange *)change toManagedObject:(NSManagedObject
    *)managedObject inSyncSession:(ISyncSession *)session
```

Parameters*coordinator*

The persistent store coordinator being synced.

change

The changes that was applied.

managedObject

The managed object that corresponds to the changes.

session

The sync session that applied the changes.

Availability

Available in Mac OS X v10.5 and later.

See Also- [persistentStoreCoordinator:willApplyChange:toManagedObject:inSyncSession:](#) (page 128)**Declared In**

ISyncCoreData.h

persistentStoreCoordinator:didCancelSyncSession:error:

Informs the receiver that a session was cancelled.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    didCancelSyncSession:(ISyncSession *)session error:(NSError *)error
```

Parameters*coordinator*

The persistent store coordinator being synced.

session

The sync session that was cancelled.

error

Describes the error that caused the cancellation.

Availability

Available in Mac OS X v10.5 and later.

See Also- [persistentStoreCoordinator:didFinishSyncSession:](#) (page 127)**Declared In**

ISyncCoreData.h

persistentStoreCoordinator:didCommitChanges:inSyncSession:

Informs the receiver that all applied changes were committed during a sync session.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    didCommitChanges:(NSDictionary *)changes inSyncSession:(ISyncSession *)session
```

Parameters*coordinator*

The persistent store coordinator being synced.

changes

A dictionary containing the changes to possibly multiple objects that were applied and committed. The dictionary contains the following keys: `NSInsertedObjectsKey`, `NSUpdatedObjectsKey`, and `NSDeletedObjectsKey`.

session

The sync session that committed the changes.

Discussion

Typically, this method is invoked after the persistent store changes are saved and the sync session receives the `clientCommittedAcceptedChanges` (page 64) message. This method can be invoked multiple times during a sync session.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncCoreData.h`

`persistentStoreCoordinator:didFinishSyncSession:`

Informs the receiver that a session was finished.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
  didFinishSyncSession:(ISyncSession *)session
```

Parameters*coordinator*

The persistent store coordinator being synced.

session

The sync session that finished.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [persistentStoreCoordinator:didCancelSyncSession:error:](#) (page 126)

Declared In

`ISyncCoreData.h`

`persistentStoreCoordinator:didPullChangesInSyncSession:`

Informs the receiver that changes were pulled from the sync engine.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
  didPullChangesInSyncSession:(ISyncSession *)session
```

Parameters*coordinator*

The persistent store coordinator being synced.

session

The sync session that pulled the changes.

Availability

Available in Mac OS X v10.5 and later.

See Also- [persistentStoreCoordinator:willPullChangesInSyncSession:](#) (page 129)**Declared In**

ISyncCoreData.h

persistentStoreCoordinator:didPushChangesInSyncSession:

Informs the receiver that client changes were pushed to the sync engine.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    didPushChangesInSyncSession:(ISyncSession *)session
```

Parameters*coordinator*

The persistent store coordinator being synced.

session

The sync session that pushed the changes.

Availability

Available in Mac OS X v10.5 and later.

See Also- [persistentStoreCoordinator:willPushChangesInSyncSession:](#) (page 130)**Declared In**

ISyncCoreData.h

persistentStoreCoordinator:willApplyChange:toManagedObject:inSyncSession:

Informs the receiver that pulled changes will be applied to a specific record during a sync session.

```
- (ISyncChange *)persistentStoreCoordinator:(NSPersistentStoreCoordinator
 *)coordinator willApplyChange:(ISyncChange *)change
    toManagedObject:(NSManagedObject *)managedObject inSyncSession:(ISyncSession
 *)session
```

Parameters*coordinator*

The persistent store coordinator being synced.

*change*The changes that will be applied. An `ISyncChange` object can represent a delete record change, as well as an insert and update record change.

managedObject

The managed object that corresponds to the changes.

session

The sync session that is applying the changes.

Return Value

The change to apply. `nil` if you do not want to apply this change.

Discussion

Implement this method if you want to modify a change before it is applied.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [persistentStoreCoordinator:didApplyChange:toManagedObject:inSyncSession:](#) (page 125)

Declared In

`ISyncCoreData.h`

persistentStoreCoordinator:willDeleteRecordWithIdentifier:inSyncSession:

Informs the receiver that a specific record will be deleted during the pushing phase of a sync session.

```
- (BOOL)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    willDeleteRecordWithIdentifier:(NSString *)identifier inSyncSession:(ISyncSession
    *)session
```

Parameters

coordinator

The persistent store coordinator being synced.

identifier

The identifier for the record that will be deleted.

session

The sync session that is pushing records.

Return Value

YES to delete the record; otherwise, NO.

Discussion

Implement this method if you want to verify if a record should be deleted before it is deleted.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncCoreData.h`

persistentStoreCoordinator:willPullChangesInSyncSession:

Informs the receiver that changes will be pulled from the sync engine.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    willPullChangesInSyncSession:(ISyncSession *)session
```

Parameters

coordinator

The persistent store coordinator being synced.

session

The sync session that is pulling the changes.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [persistentStoreCoordinator:didPullChangesInSyncSession:](#) (page 127)

Declared In

ISyncCoreData.h

persistentStoreCoordinator:willPushChangesInSyncSession:

Informs the receiver that client changes will be pushed to the sync engine.

```
- (void)persistentStoreCoordinator:(NSPersistentStoreCoordinator *)coordinator
    willPushChangesInSyncSession:(ISyncSession *)session
```

Parameters

coordinator

The persistent store coordinator being synced.

session

The sync session that is pushing the changes.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [persistentStoreCoordinator:didPushChangesInSyncSession:](#) (page 128)

Declared In

ISyncCoreData.h

persistentStoreCoordinator:willPushRecord:forManagedObject:inSyncSession:

Informs the receiver that client changes to a specific record will be pushed to the sync engine.

```
- (NSDictionary *)persistentStoreCoordinator:(NSPersistentStoreCoordinator
 *)coordinator willPushRecord:(NSDictionary *)record
    forManagedObject:(NSManagedObject *)managedObject inSyncSession:(ISyncSession
 *)session
```

Parameters

coordinator

The persistent store coordinator being synced.

record

The record that will be pushed.

managedObject

The managed object that corresponds to the record.

session

The sync session that is pushing records.

Return Value

The record to push. `nil` if you do not want to push the record.

Discussion

Implement this method if you want to modify a record before pushing it.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncCoreData.h`

persistentStoreCoordinatorShouldStartSyncing:

Returns whether or not the persistent store coordinator should start syncing.

```
- (BOOL)persistentStoreCoordinatorShouldStartSyncing:(NSPersistentStoreCoordinator
*)coordinator
```

Parameters

coordinator

The persistent store coordinator being synced.

Return Value

YES if the persistent store coordinator can start syncing; otherwise, NO.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`ISyncCoreData.h`

Document Revision History

This table describes the changes to *Sync Services Framework Reference*.

Date	Notes
2007-10-31	Added ISyncUIHelper.
2007-07-11	Updated for Mac OS X v10.5. Changed title to "Sync Services Framework Reference."

REVISION HISTORY

Document Revision History

Index

A

`addRequestMode`: **instance method** [37](#)
`applyChange:forEntityName:`
 `remappedRecordIdentifier:formattedRecord:error:`
 protocol instance method [109](#)
`attributedStringForIdentityPropertiesWithNames:`
 `inRecord:comparisonRecords:firstLineAttributes:`
 `secondLineAttributes`: **protocol instance method**
 [120](#)
`attributedStringForPropertiesWithNames:inRecord:`
 `comparisonRecords:defaultAttributes:`
 protocol instance method [121](#)

B

`beginTransactionInBackgroundWithClient:entityNames:`
 `target:selector`: **class method** [55](#)
`beginTransactionInBackgroundWithClient:entityNames:`
 `target:selector:lastAnchors`: **class method**
 [56](#)
`beginTransactionWithClient:entityNames:beforeDate:`
 class method [58](#)
`beginTransactionWithClient:entityNames:beforeDate:`
 `lastAnchors`: **class method** [59](#)

C

`cancelPreviousBeginSessionWithClient`: **class method** [60](#)
`cancelSyncing` **instance method** [61](#)
`canPullChangesForEntityName`: **instance method** [20](#)
`canPushChangesForEntityName`: **instance method** [20](#)
`changedRecordsForEntityName:moreComing:error:`
 protocol instance method [110](#)
`changeEnumeratorForEntityNames`: **instance method**
 [61](#)
`changes` **instance method** [13](#)

`changesForEntityName:moreComing:error`: **protocol instance method** [111](#)
`changeWithType:recordIdentifier:changes`: **class method** [12](#)
`client` **instance method** [80](#)
`clientAcceptedChangesForRecordWithIdentifier:`
 `formattedRecord:newRecordIdentifier:`
 instance method [62](#)
`clientChangedRecordIdentifiers`: **instance method**
 [63](#)
`clientCommittedAcceptedChanges` **instance method**
 [64](#)
`clientCommittedAcceptedChangesWithNextAnchors:`
 instance method [64](#)
`clientDescriptionURL` **protocol instance method** [112](#)
`clientDidResetEntityNames`: **instance method** [65](#)
`clientFinishedPushingChangesWithNextAnchors:`
 instance method [66](#)
`clientIdentifier` **instance method** [21](#)
`clientIdentifier` **protocol instance method** [112](#)
`clientInfoForRecordWithIdentifier`: **instance method** [66](#)
`clientLostRecordWithIdentifier:`
 `shouldReplaceOnNextSync`: **instance method** [67](#)
`clientRefusedChangesForRecordWithIdentifier:`
 instance method [67](#)
`clientType` **instance method** [21](#)
`clientWantsToPushAllRecordsForEntityNames:`
 instance method [68](#)
`clientWithIdentifier`: **instance method** [38](#)

D

`dataSource` **instance method** [80](#)
`delegate` **instance method** [80](#)
`deleteAllRecordsForEntityName:error`: **protocol instance method** [112](#)
`deleteRecordWithIdentifier`: **instance method** [68](#)
`displayName` **instance method** [21](#)

E

enabledEntityNames **instance method** 22
 entityNamesToPull **protocol instance method** 113
 entityNamesToSync **protocol instance method** 113
 Error Codes 97

F

filterMatchingAllFilters: **class method** 34
 filterMatchingAtLeastOneFilter: **class method** 34
 filters **instance method** 22
 finishSyncing **instance method** 69, 81

H

handlesSyncAlerts **instance method** 81

I

identifiersForRecordsToDeleteForEntityName:
 moreComing:error: **protocol instance method** 114
 imagePath **instance method** 22
 initWithChangeType:recordIdentifier:changes:
instance method 13
 isCancelled **instance method** 69
 isEnabled **instance method** 38
 isEnabledForEntityName: **instance method** 23
 isEqual: **protocol instance method** 104
 ISyncAvailabilityChangedNotification
 notification 42
 ISyncChangePropertyActionKey **constant** 15
 ISyncChangePropertyClear **constant** 16
 ISyncChangePropertyNameKey **constant** 16
 ISyncChangePropertySet **constant** 16
 ISyncChangePropertyValueKey **constant** 16
 ISyncChangeTypeAdd **constant** 15
 ISyncChangeTypeDelete **constant** 15
 ISyncChangeTypeModify **constant** 15
 ISyncChange—Change Types 95
 ISyncChange—Property Keys 98
 ISyncClientTypeApplication **constant** 31
 ISyncClientTypeDevice **constant** 31
 ISyncClientTypePeer **constant** 31
 ISyncClientTypeServer **constant** 31
 ISyncClient—Client Status 96
 ISyncClient—Client Types 98

ISyncErrorDomain 99

ISyncInvalidEntityException **constant** 76
 ISyncInvalidRecordException **constant** 76
 ISyncInvalidRecordIdentifiersKey **constant** 76
 ISyncInvalidRecordReasonsKey **constant** 76
 ISyncInvalidRecordsKey **constant** 76

ISyncManager—Exceptions 98

ISyncRecordEntityNameKey **constant** 75
 ISyncServerUnavailableException **constant** 42
 ISyncSessionCancelledException **constant** 76
 ISyncSessionClientAlreadySyncingError **constant** 97
 ISyncSessionDriverChangeAccepted **constant** 118
 ISyncSessionDriverChangeError **constant** 118
 ISyncSessionDriverChangeIgnored **constant** 118
 ISyncSessionDriverChangeRefused **constant** 118
 ISyncSessionDriverFatalError **constant** 97
 ISyncSessionDriverModeFast **constant** 118
 ISyncSessionDriverModeRefresh **constant** 118
 ISyncSessionDriverModeSlow **constant** 118
 ISyncSessionDriverPullFailureError **constant** 97
 ISyncSessionDriverRegistrationError **constant** 97

ISyncSessionDriver—Preferred Sync Mode 96**ISyncSessionDriver—Status of Applying Changes** 96

ISyncSessionUnavailableException **constant** 76
 ISyncSessionUserCanceledSessionError **constant** 97

ISyncSession—Exceptions 99**ISyncSession—Key** 99

ISyncStatusCancelled **constant** 32
 ISyncStatusErrors **constant** 32
 ISyncStatusFailed **constant** 32
 ISyncStatusNever **constant** 32
 ISyncStatusRunning **constant** 32
 ISyncStatusSuccess **constant** 32
 ISyncStatusWarnings **constant** 32
 ISyncUnsupportedEntityException **constant** 76

L

lastAnchorForEntityName: **protocol instance method** 114
 lastError **instance method** 81
 lastSyncDateForEntityName: **instance method** 23
 lastSyncStatusForEntityName: **instance method** 24

M

managedObjectContextsToMonitorWhenSyncing-
PersistentStoreCoordinator: **protocol instance
method 124**

managedObjectContextsToReloadAfterSyncing-
PersistentStoreCoordinator: **protocol instance
method 125**

N

nextAnchorForEntityName: **protocol instance method
115**

O

objectForKey: **instance method 24**

P

persistentStoreCoordinator:didApplyChange:
toManagedObject:inSyncSession: **protocol
instance method 125**

persistentStoreCoordinator:didCancelSyncSession:
error: **protocol instance method 126**

persistentStoreCoordinator:didCommitChanges:
inSyncSession: **protocol instance method 126**

persistentStoreCoordinator:didFinishSyncSession:
protocol instance method 127

persistentStoreCoordinator:
didPullChangesInSyncSession: **protocol
instance method 127**

persistentStoreCoordinator:
didPushChangesInSyncSession: **protocol
instance method 128**

persistentStoreCoordinator:willApplyChange:
toManagedObject:inSyncSession: **protocol
instance method 128**

persistentStoreCoordinator:
willDeleteRecordWithIdentifier:inSyncSession:
protocol instance method 129

persistentStoreCoordinator:
willPullChangesInSyncSession: **protocol
instance method 129**

persistentStoreCoordinator:
willPushChangesInSyncSession: **protocol
instance method 130**

persistentStoreCoordinator:willPushRecord:
forManagedObject:inSyncSession: **protocol
instance method 130**

persistentStoreCoordinatorShouldStartSyncing:
protocol instance method 131

preferredSyncModeForEntityName: **protocol instance
method 115**

prepareToPullChangesForEntityNames:beforeDate:
instance method 70

prepareToPullChangesInBackgroundForEntityNames:
target:selector: **instance method 71**

pushChange: **instance method 71**

pushChangesFromRecord:withIdentifier: **instance
method 72**

R

record **instance method 14**

recordIdentifier **instance method 14**

recordIdentifierForReference:isModified:
instance method 46

recordReferenceForRecordWithIdentifier:
instance method 47

recordsForEntityName:moreComing:error: **protocol
instance method 116**

recordsWithIdentifiers: **instance method 48**

recordsWithMatchingAttributes: **instance method
48**

registerClientWithIdentifier:descriptionFilePath:
instance method 38

registerSchemaWithBundlePath: **instance method
39**

removeRequestMode: **instance method 40**

requestModes **instance method 40**

S

schemaBundleURLs **protocol instance method 117**

session **instance method 82**

sessionBeginTimeout **protocol instance method 117**

sessionDriver:didPullAndReturnError:
<NSObject> **delegate method 84**

sessionDriver:didPushAndReturnError:
<NSObject> **delegate method 85**

sessionDriver:didRegisterClientAndReturnError:
<NSObject> **delegate method 85**

sessionDriver:willFinishSessionAndReturnError:
<NSObject> **delegate method 85**

sessionDriver:willPullAndReturnError:
<NSObject> **delegate method 86**

sessionDriver:willPushAndReturnError:
 <NSObject> delegate method [86](#)
 sessionDriverDidCancelSession: <NSObject>
 delegate method [86](#)
 sessionDriverDidFinishSession: <NSObject>
 delegate method [86](#)
 sessionDriverWillCancelSession: <NSObject>
 delegate method [87](#)
 sessionDriverWithDataSource: class method [79](#)
 sessionPullChangesTimeout protocol instance
 method [117](#)
 setClientInfo:forRecordWithIdentifier: instance
 method [73](#)
 setDelegate: instance method [82](#)
 setDisplayName: instance method [24](#)
 setEnabled:forEntityNames: instance method [25](#)
 setFilters: instance method [25](#)
 setHandlesSyncAlerts: instance method [83](#)
 setImagePath: instance method [26](#)
 setObject:forKey: instance method [26](#)
 setShouldReplaceClientRecords:forEntityNames:
 instance method [27](#)
 setShouldSynchronize:withClientsOfType:
 instance method [27](#)
 setStoresFastSyncDetailsAtURL:forPersistentStore:
 instance method [90](#)
 setSyncAlertHandler:selector: instance method
 [28](#)
 setSyncAlertToolPath: instance method [29](#)
 sharedManager class method [37](#)
 shouldApplyRecord:withRecordIdentifier:
 protocol instance method [104](#)
 shouldPullChangesForEntityName: instance method
 [73](#)
 shouldPushAllRecordsForEntityName: instance
 method [73](#)
 shouldPushChangesForEntityName: instance method
 [74](#)
 shouldReplaceAllRecordsOnClientForEntityName:
 instance method [74](#)
 shouldReplaceClientRecordsForEntityName:
 instance method [30](#)
 shouldSynchronizeWithClientsOfType: instance
 method [30](#)
 snapshotOfRecordsInTruth instance method [75](#)
 snapshotOfRecordsInTruthWithEntityNames:
 usingIdentifiersForClient: instance method
 [40](#)
 sourceIdentifiersForRelationshipName:
 withTargetIdentifier: instance method [48](#)
 startAsynchronousSync: instance method [83](#)
 supportedEntityNames instance method [30](#)
 supportedEntityNames protocol instance method [105](#)

sync instance method [84](#)
 syncAlertToolPath instance method [31](#)
 syncWithClient:inBackground:handler:error:
 instance method [90](#)

T

targetIdentifiersForRelationshipName:
 withSourceIdentifier: instance method [49](#)
 type instance method [14](#)

U

unregisterClient: instance method [41](#)
 unregisterSchemaWithName: instance method [41](#)