Sync Services Tutorial

Cocoa > Syncing

2007-07-11

Contents

Tables

Introduction to Sync Services Tutorial

Important: This is a preliminary document about using Syncrospector, a tool that is in development. Although this document has been reviewed for technical accuracy, it is not final. Apple Computer is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein. This information is subject to change. Newer versions of this document may be provided with future releases. For information about updates to this and other developer documentation, view the New & Updated sidebars in subsequent releases of the Reference Library.

Syncrospector is a development tool used to help debug your Sync Services applications. Sync Services is a framework containing all the components you need to sync your applications and devices. By using Sync Services, user data can be synced with other applications and devices on the same computer, or other computers over the network via .Mac.

For many reasons, designing, implementing, and debugging your syncing application can be challenging—for example, multiple applications may join a sync session, your application may crash, your local data file may be missing when your application launches, and you might want to trickle sync. This tutorial focuses on the responsibilities of the application and introduces an invaluable debugging tool.

This tutorial uses two example applications, SimpleStickies and FancyStickies, to show you how to debug your applications and implement additional features. The sample code, called StickiesExample, is available on the ADC Reference Library in:

ADC Home > Reference Library > Documentation > Apple Applications > iSync

Who Should Read This Document?

You should read this document if you are developing an application that uses Sync Services. This document contains information on developer tools and techniques for debugging your Sync Services application.

Organization of This Document

The following articles cover several aspects of developing a Sync Services application:

- "Using Syncrospector to Debug Your Application" (page 9) explains how to set up the Syncrospector tool and view status about clients, the truth database, and sync sessions.
- "Viewing Sync Session Results" (page 11) describes how to view the call history of a sync session.
- "Syncing Multiple Applications" (page 15) describes how to view the call history of two clients that join the same sync session.
- "Trickle Syncing" (page 19) describes how to implement trickle syncing.

- "Refresh Syncing" (page 21) describes how to modify a client to refresh sync when local records have been inadvertently removed.

See Also

For a complete description of the Sync Services API, read:

- *Sync Services Programming Guide* discusses Sync Services concepts such as the architecture, components, creating a schema, and managing a sync session.
- *Sync Services Framework Reference* contains a description of each class, method and type.

If you are using or extending an Apple Applications schema, such as contacts, bookmarks or calendars, read:

- *Apple Applications Schema Reference* describes the different Apple Applications schemas that you can use in your applications.

The `/Developer/Examples/Sync Services` folder contains more in-depth sample code.

Using Syncrospector to Debug Your Application

Syncrospector is a debugging tool that you can use to inspect clients, the truth database, and the call history of sync sessions. This article uses the sample application, SimpleStickies, to demonstrate how to use Syncrospector. This article gives you a quick tour of Syncrospector and describes the basic debugging steps.

Building and Running SimpleStickies

Build and run the SimpleStickies application to create some sample content for this tutorial.

1. Open `SimpleStickies.xcode`, located in the `StickiesExample/Examples/SimpleStickies1_TrickleSyncing` folder.
2. Click Build and Go in Xcode to launch the application.

SimpleStickies registers the Stickies schema and the SimpleStickies client. You can examine these changes to the state of the sync engine using Syncrospector.

Viewing Sync Clients

You can use Syncrospector to see if your client successfully registered with the sync engine.

1. Launch Syncrospector, located in the `StickiesExample/Applications` folder.
2. Select Clients from the left-side pull-down menu to see the registered clients.

   You should see SimpleStickies listed as one of the registered clients. Most of the client properties displayed by Syncrospector (described in Table 1 (page 9))are set in the client description property list.

Table 1 Client Properties

Property	Description
Image	If you specified an icon in the client description, it appears in the first column and in the detail view if you select a client. The icon and display name are used in alert panels.
Display Name	The display name should be a user-friendly name of for the client.
Client Identifier	The client identifier needs to be unique, so reverse DNS-style naming is recommended.

Property	Description
Type	The type of client (app, server, or device) is displayed in the Type column. Most clients are of type "app" meaning an application. .Mac is an example of a server and an iPod is an example of a device. If .Mac is turned on, you see it listed as a server.
Last Sync	The last column displays the date the client last synced.

Viewing Sync Session Results

You can also use Syncrospector to view the results of a sync session—for example, you can view the changes to the truth database and even the call history of a sync session. Before you can view this information, you need to set some preferences in Syncrospector.

Setting Syncrospector Preferences

The following preferences are useful when using Syncrospector to debug your applications. You will need to set these preferences before syncing your application to examine the call history.

1. Launch Syncrospector, located in the `StickiesExample/Applications` folder.
2. Select Preferences from the Syncrospector menu.
3. Select "Show status in menu bar."
4. Select "Extensive logging."
5. Select "SyncServices.framework call history."
6. Restart SimpleStickies.

Note: When you are done debugging, you can remove the `save.isyncplayback` and `syncservices.log` files located in `~/Library/Application Support/SyncServices/Local`. Remove the log files only after "Extensive logging" and "SyncServices.framework call history" are turned off in the Syncrospector Preferences panel and Syncrospector is no longer running.

Syncing SimpleStickies

Add and sync a few note records using SimpleStickies as follows:

1. Launch SimpleStickies to create a few note records by clicking the Add button below the table.
2. Select a note record in the table, and set a few properties of the note records in the detail interface—for example, enter some text in the Text field, and change the color of each note record.
3. Select Sync from the File menu to save a local copy of the records and push them to the sync engine.
4. If a panel appears asking whether you would like to sync SimpleStickies with other applications—for example, FancyStickies—click Allow.

A panel may appear if you skipped ahead and ran FancyStickies. When you first sync a client that shares a schema with another client, a panel appears asking whether you want to sync with other clients. If you click Allow, the panel does not appear again unless you unregister the client.

Viewing the Client State

First, you should check to see that the client synced and that it is in the right state for the next sync.

1. Launch Syncrospector, and select Clients from the left-side pull-down menu. Notice that the Last Sync date for SimpleStickies changed.
2. Select SimpleStickies from the table, and click Sync State in the detail view. The detail view shows that SimpleStickies successfully synced the `com.mycompany.stickies.Note` entity and is expected to fast sync the next time.

Read *Sync Services Programming Guide* for details on the negotiation phase and sync modes.

Viewing the Truth Database

You can also check to see whether your records were pushed. To do this, examine the contents of the truth database using Syncrospector.

1. Using Syncrospector, select Truth from the left-side pull-down menu.
2. Click Reload if Syncrospector was running when you synced and the table is empty.
3. Display the Note entities by entering `com.mycompany.stickies.Note` in the search field.

   The records you created should appear in the table.
4. Select a record to view details.
5. Click Properties in the detail view to view the record's property values.
6. Click Client States to view the different clients that synced this record.

   Note that the truth database uses a global record identifier that is different from the SimpleStickies local record identifier. Local identifiers are different for each client.

Viewing the Call History

You can also use Syncrospector to view the call history of a sync session.

1. Using Syncrospector, select History from the left-side pull-down menu.

2. Select the last sync session from the list and click the outline triangle.

   The list expands to show the clients that participated in that sync session.

3. Select the `com.mycompany.SimpleStickies` client from the list.

   If other clients have joined the sync session, they are also listed here. If changes were pushed to the sync engine, they are listed in the detail view below.

4. While the client is selected, click Open Call History to see the history of messages sent from the client to the sync engine during that session.

For a complete description of sync session states, read *Sync Services Programming Guide*. For a description of each method, read *Sync Services Framework Reference*.

Syncing Multiple Applications

The sync engine allows multiple applications that use the same schema to sync together. When one client begins a sync session, other clients may be invited to join the same sync session. Since SimpleStickies and FancyStickies share the same schema, each is given the opportunity to join the sync session started by the other client. You can use Syncrospector to view the results of multiple clients syncing together.

Displaying Change Alerts

When debugging multiple clients, it is useful to turn on the change alert panels. Normally, change alert panels are displayed for .Mac only. You can turn on change alert panels for all clients by using Syncrospector.

1. Launch Syncrospector.
2. Select Preferences from the Syncrospector menu to open the Preferences panel.
3. Click Always Show Data Change Alert.

Launching FancyStickies

Launch FancyStickies to register a second client that uses the Stickies schema.

1. Launch FancyStickies, located in the `StickiesExample/Applications` folder.
2. If a panel appears asking whether you want to sync FancyStickies with other applications—for example, to SimpleStickies—click Allow.

   FancyStickies syncs at startup so you should see a sticky window appear for each Note record you created using SimpleStickies.
3. Using Syncrospector, select Clients from the left-side menu to view the state of the FancyStickies client.
4. Click History, open the last sync session, select `com.mycompany.FancyStickies`, and select Open Call History to view the sync session details.

Syncing FancyStickies and SimpleStickies

Now make some changes in either application and sync both applications to push changes from one application to the other.

1. Using FancyStickies, change the text in the sticky window, and select Save from the File menu.

   FancyStickies saves and syncs the changes—pushing the changes to the sync engine.

2. If change alert panels are turned on, a Sync Alert panel appears. Check to see that it detected the correct number of changes you made and then click Allow.

3. Using SimpleStickies, select Sync from the File menu to pull the changes from the sync engine.

4. Click Allow if a Sync Alert panel appears.

5. Using SimpleStickies, change some attributes of Note records, or add and remove some records. Select Sync from the File menu again to push the changes.

   FancyStickies joins the sync session and pulls the changes automatically.

6. If a change alert panel appears, click Allow.

Read "Trickle Syncing" (page 19)to learn how to configure SimpleStickies to join a sync session when FancyStickies syncs.

Viewing the Session History of Multiple Clients

Use Syncrospector to view the call history of multiple clients that join the same sync session.

1. Using Syncrospector, select Clients from the pull-down menu to check the sync state of each client.

   The Last Sync date for SimpleStickies and FancyStickies should be the same.

2. Select History from the pull-down menu and optionally, click Reload.

3. Select the last sync session from the table and open the clients.

   You should see both SimpleStickies and FancyStickies listed as clients that joined the last sync session.

4. Select `com.mycompany.FancyStickies` and click Open Call History, and then select `com.mycompany.SimpleStickies` and click Open Call History to compare the history of each client in separate windows.

5. For example, click the `pushChangesFromRecord:withIdentifier:` method in the SimpleStickies Call History window to see the records that SimpleStickies pushed to the sync engine.

6. Click the `nextObject` method in the FancyStickies Call History window to see the records that FancyStickies pulled from the sync engine.

Viewing Sync Plans in Action

You can also view the call history of multiple clients during a sync session.

1. Using Syncrospector, select Show Sync Plans from the Window menu.
2. Again, modify some records using SimpleStickies—for example, change the colors of some notes—and select Sync from the File menu to push the changes.
3. If a Sync Alert panel appears in FancyStickies, click Allow to continue.
4. View the sync plans in the Sync Server window in Syncrospector by selecting the last sync session.

   SimpleStickies and FancyStickies should be listed in the Participants column in the detail view. The Entities column should show the entities each client synced—for example, `com.mycompany.stickies.Note`.

Trickle Syncing

Trickle syncing is a common style or behavior that syncing clients exhibit. When your application trickle syncs, changes are synced automatically and frequently in the background. Consequently, user data appears when and where users want it.

An application that trickle syncs should do the following:

- Sync after launching
- Sync before termination
- Sync after saving
- Join sync sessions

FancyStickies already trickle syncs with the exception of saving automatically—you have to select Save from the File menu to save records. Since FancyStickies syncs after every save action, saving changes also pushes them to the sync engine.

Currently, SimpleStickies saves automatically whenever the user makes changes. SimpleStickies also syncs after launching and on command—for example, you select Sync from the File menu to sync SimpleStickies.

However, SimpleStickies doesn't join any sync sessions—specifically, it should sync whenever FancyStickies syncs. This article explains how to modify SimpleStickies to sync with other clients.

Setting Alert Handlers

If you want your client to join sync sessions, you need to tell the sync engine what types of clients you are interested in and set an alert handler as follows.

1. Open the Xcode project file, located in the `StickiesExample/Examples/SimpleStickies1...` folder, and add a `setSyncAlertHandler:selector:` method to SyncIt. Add the following method implementation to `SyncIt.m`:

```
-(void)setSyncAlertHandler:(id)handler selector:(SEL)selector
{
    ISyncClient* client = [self _syncClient];
    if (client == nil) {
        NSLog(@"Warning: Can't get sync client to register alert handler");
        return;
    }
    [client setShouldSynchronize:YES
withClientsOfType:ISyncClientTypeApplication];
    [client setShouldSynchronize:YES withClientsOfType:ISyncClientTypeServer];
    [client setSyncAlertHandler:handler selector:selector];
}
```

Before setting the alert handler, the SyncIt object tells the ISyncClient object that the client is interested in syncing with both applications and servers—for example, .Mac.

2. Add the following method declaration to `SyncIt.h`:

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

3. Modify the `init` method in `AppDelegate.m` to register an alert handler method after creating the SyncIt object. Add this line of code after registering the schemas:

```
[_syncIt setSyncAlertHandler:self
selector:@selector(_client:mightWantToSyncEntityNames:)];
```

4. Now implement the handler, `_client:mightWantToSyncEntityNames:`, to join the sync session.

```
- (void)_client:(ISyncClient *)client mightWantToSyncEntityNames:(NSArray
*)entityNames
{
    [self syncAction:self];
}
```

5. Add the following method declaration to the AppDelegatePrivate category in `AppDelegate.m`:

```
- (void)_client:(ISyncClient *)client mightWantToSyncEntityNames:(NSArray
*)entityNames;
```

6. Click Build and Go in Xcode to test the application—for example, make some changes in FancyStickies and sync it to see whether SimpleStickies joins the sync session.

Refresh Syncing

Your application needs to be robust enough to recover from unexpected events—for example, the absence of a local file. Suppose the user creates some data and syncs it, and for some unknown reason, removes the local file. At this point, the sync engine assumes that the client has copies of the records and will not send them to the client on the next sync. The client needs to explicitly request a refresh sync when this occurs; otherwise, records may be destroyed.

Accidently Deleting Records

For example, if you create some records using SimpleStickies and remove the local file, you can lose all your records. Follow these steps to test if your application properly handles a missing local file.

1. If you haven't created any records with SimpleStickies, launch SimpleStickies, create some records, save (and sync) the file, and quit.

2. Remove the `com.mycompany.SimpleStickies.xml` file, located in the `~/Library/Application Support/SyncExamples` folder, which contains the local copy of the client's records.

3. Using Syncrospector, select Clients from the pull-down menu to view the state of SimpleStickies—select SimpleStickies from the table and click Sync State in the detail view.

   If SimpleStickies is expected to fast sync the next time it syncs, and it fails to load the local records when launching, it will fail to apply any changes pulled from the sync engine, except adds. If the application slow syncs the next time, SimpleStickies will not push any records, and the sync engine will assume that the client deleted those records. This may result in the unintentional loss of data.

4. Force SimpleStickies to slow sync by selecting SimpleStickies from the list of clients and selecting Slow Sync from the right-hand side pull-down menu.

   When you click Sync State the Will Sync mode should read "slow."

5. Launch SimpleStickies again.

   SimpleStickies syncs immediately after launching. If FancyStickies is running, it will join the sync session.

6. If a change alert panel appears in FancyStickies, click Allow to continue.

   Because SimpleStickies doesn't push any records, the sync engine issues delete changes to FancyStickies and all its sticky windows disappear.

7. Examine the Note records in the truth database by selecting Truth from the left-side pull-down menu in Syncrospector. The sync engine deleted all of the Note records.

   You will see that all the Note records have been marked for deletion.

To avoid this from happening in your application, you should request a refresh sync if you detect that the local file is missing—that is, remove all local records and "pull the truth."

Requesting a Refresh Sync

The AppDelegate `dataSource` accessor method creates its data source by attempting to load the data from a file. You need to modify this implementation to set the preferred sync mode to refresh when the backup file doesn't exist. You do this by sending `setPreferredSyncMode:` to the SyncIt object, passing `RefreshSyncMode` as the argument. Add this line to the `dataSource` method after the `NSLog()` function call:

```
        [_syncIt setPreferredSyncMode:RefreshSyncMode];
```

The code fragment should now look like this:

```
    NSURL *url = [NSURL fileURLWithPath:[applicationSupportFolder
stringByAppendingPathComponent:[self fileName]]];
    if ([self _readFromURL:url] == NO) {
        NSLog(@"No data from backup file on init, creating empty new data
source");
        [_syncIt setPreferredSyncMode:RefreshSyncMode];
    }
    else {
        NSLog(@"Read in some data on init from backup file");
    }
```

Now examine the `SyncIt.m` code to see how the preferred sync mode is used when syncing your application.

1. Open `SyncIt.m` in Xcode.

2. Search for `RefreshSyncMode`, and you will see this code fragment:

```
        if (_syncMode == RefreshSyncMode) {
            NSLog(@"Requesting refresh sync from engine for entity names %@",
    [_syncSource entityNames]);
            [_syncSession clientDidResetEntityNames:[_syncSource entityNames]];
        }
```

3. Build and test the application. Create some records using SimpleStickies and then quit the application. Remove the local file and click Build and Go in Xcode to restart SimpleStickies.

Sending `clientDidResetEntityNames:` to the sync session notifies the sync engine that the client's sync state should be reset. Consequently, the client will not pull delete changes for the records that it lost.

Follow the steps in "Accidently Deleting Records" (page 21)to test if the changes to SimpleStickies, requesting a refresh sync, works.

Document Revision History

This table describes the changes to *Sync Services Tutorial*.

Date	Notes
2007-07-11	Minor editorial corrections throughout.
2005-07-07	Minor editorial corrections throughout.
2005-06-04	New document that explains how to debug multiple syncing applications using the Syncrospector development tool.