6

Creating Sync Application for iOS

The following sections describe how to use mobile client sync package for iOS to develop mobile sync application for Sqlite or Berkeley DB:

- Section 6.1, "Contents of the iOS Mobile Client Package"
- Section 6.2, "Building Sync Application"

You need to download and install the mobile sync package on your Mac from the mobile server in order to develop the application.

Refer to Section 2.3.7, "Installing iOS Mobile Client" for requirements and instructions.

6.1 Contents of the iOS Mobile Client Package

Here is the directory tree for Berkeley DB sync package:

osync-bdb-ios-11.3

doc

mcguide.pdf

include

bgmsg.h

bgsync.h

ose.h

oseerr.h

sqlitePluginErr.h

lib

ios6.0

armv7

libdb_sql-6.0.a

libosync_bdb.a

ios6.0simulator

i386

libdb_sql-6.0.a

libosync_bdb.a

res

osync_res

del.proj

osync.strings

en.lproj

osync.strings

es.lproj

osync.strings

fr.lproj

osync.strings

it.lproj

osync.strings

ja.lproj

osync.strings

ko.lproj

osync.strings

pt-BR.lproj

osync.strings

zh-Hans.lproj

osync.strings

The structure of Sqlite sync package is almost identical, except that it does not contain Berkeley DB library libdb_sql-6.0.a (instead the Sqlite library provided by the system is used):

osync-ios-11.3

doc

mcguide.pdf

include

bgmsg.h

bgsync.h

ose.h

oseerr.h

sqlitePluginErr.h

lib

ios6.0

armv7

libosync.a

ios6.0simulator

i386

libosync.a

res

osync_res

de.lproj

osync.strings

en.lproj

osync.strings

es.lproj

osync.strings

fr.lproj

osync.strings

it.lproj

osync.strings

ja.lproj

osync.strings

ko.lproj

osync.strings

pt-BR.lproj

osync.strings

zh-Hans.lproj

osync.strings

Note: The contents are subject to change when new versions/platforms are supported.

The following sections provide information on the top-level directories in the package:

- Section 6.1.1, "doc Directory"
- Section 6.1.2, "include Directory"
- Section 6.1.3, "lib Directory"
- Section 6.1.4, "res Directory"

6.1.1 doc Directory

Contains this document and the related files.

6.1.2 include Directory

Contains sync header files. Your application needs to include these files for using the sync APIs:

- **ose.h** contains foreground sync APIs and related constants and structures.
- **bgsync.h** contains background sync APIs and related constants and structures.
- **oseerr.h** contains foreground sync error code constants.

- **sqlitePluginErr.h** contains sqlite and Berkeley DB plugin error code constants that provide more information about database-related sync errors.
- **bgmsg.h** contains background sync error code constants and background sync message and other related constants.

6.1.3 lib Directory

Contains sync and Berkeley DB static libraries (for Berkeley DB client). Your application needs to link with **libosync.a** and **libosync_bdb.a** to use the sync APIs for Sqlite and Berkeley DB clients respectively. For Berkeley DB client, your application also needs to link with Berkeley DB library **libdb_sql-6.0 a** to access Berkeley DB client databases.

The contents are arranged by platform and CPU type. Both device and simulator builds are provided.

Currently iOS 6.0 and above targets are supported built for **armv7** architecture.

6.1.4 res Directory

Contains a bundle directory **osync_res** that contains resources used by sync client. Currently **osync_res** contains only string files used for sync client error messages and background sync messages.

However, more resources may be added in future to support UI components. Internationalization is currently provided for 9 languages: English, Spanish, French, German, Italian, Korean, Japanese, Portuguese(Brazilian) and Chinese Simplified.

6.2 Building Sync Application

The following sections provide information on how to build sync application:

- Section 6.2.1, "Prerequisites"
- Section 6.2.2, "Build Settings"

6.2.1 Prerequisites

This section assumes that you are already familiar with how to create iOS applications and hence describes how to include sync functionality into your iOS application project.

Note: For information on Mobile Server and Sync concepts, refer to Chapter 2, "*Synchronization*" of the "*Mobile Server Developer's Guide*".

Note: iOS client supports only native C APIs. Refer to Section 3.1.1.2, "*OSE Synchronization APIs For Native Applications*" of the *"Mobile Server Developer's Guide* " for OSE Synchronization APIs (foreground sync) and Section 3.2.1.2, "*Native APIs for the Sync Agent and Automatic Synchronization*" of the *"Mobile Server Developer's Guide"* for Automatic Synchronization APIs.

You need the following installed on your Mac:

- OS X v10.7 Lion or later version, currently OS X v.10.8 Mountain Lion.

- Xcode 4.5 or later version, currently 4.6. Xcode already includes iOS SDKs. To develop for iOS 6.1 and later Xcode 4.6 is required (it includes iOS 6.1 SDK).
- Mobile Sync package described in Section 6.1, "Contents of the iOS Mobile Client Package".

6.2.2 Build Settings

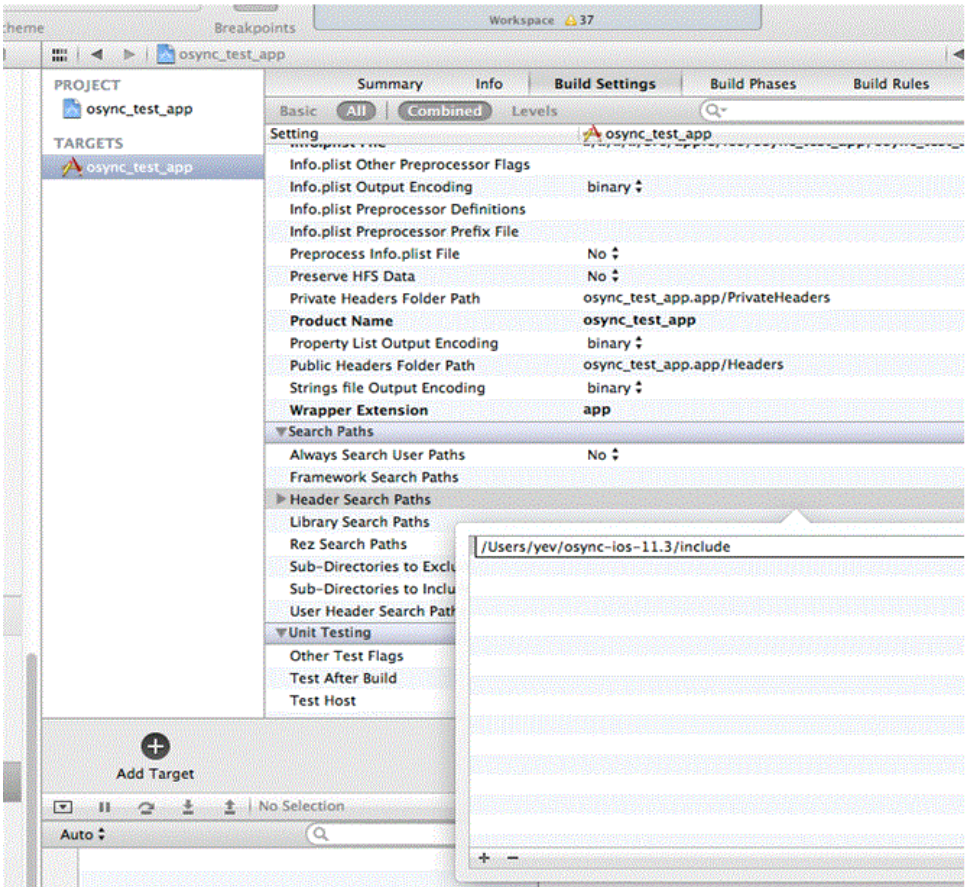
This section assumes that you are already familiar with how to create iOS applications and hence describes how to include sync functionality into your iOS application project. For more information see the following section:

- Section 6.2.2.1, "Header Search Path"

6.2.2.1 Header Search Path

- Click on your project, then on your application target, then on "**Build Phases**" tab.
- Scroll down until you see "**Search Paths**" section and double-click on a line of "**Header Search Paths**".
- A window will pop up. Click "**+**" in the bottom left corner.
- Add "**include**" directory from the sync package to the header search path (either via absolute path or relative path from your project directory).

Figure 6–1 Add "include" Directory to Header Search Path



Note: In order to use Sqlite C APIs to access your sync application databases, you also need to include "**sqlite3.h**" header file into your source files.

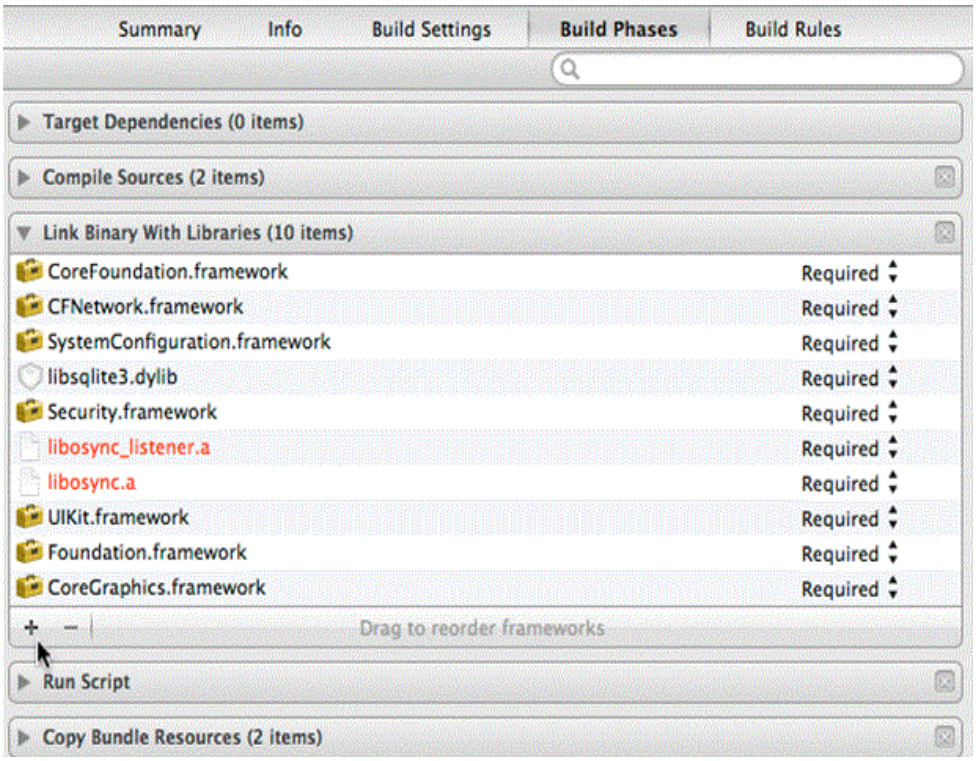
This file is already provided by iOS SDK so no additional include path is necessary. Berkeley DB fully supports Sqlite C APIs, so you can use "**sqlite3.h**" for Berkeley DB clients as well.

Refer to Berkeley DB documentation for more details.

6.2.3 Link With Libraries

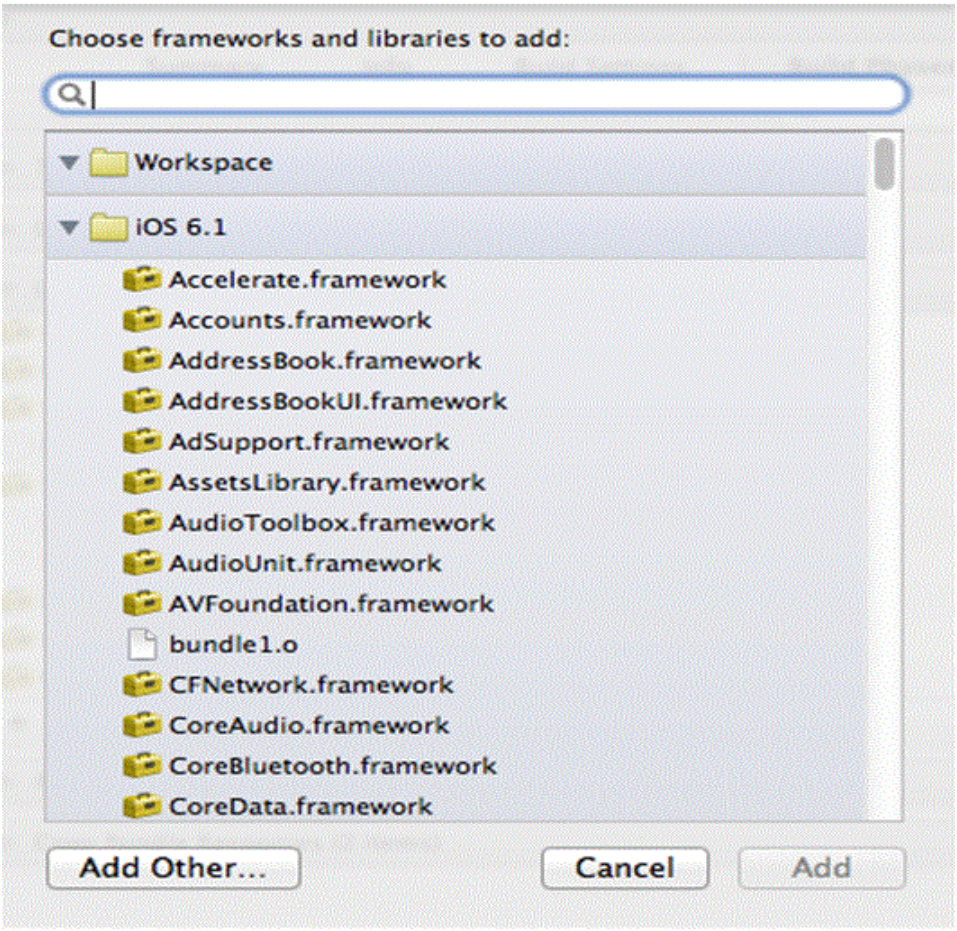
On the same screen, click "**Build Phases**" tab and toggle "**Link Binary With Libraries**". Click on "**+**".

Figure 6–2 Link With Libraries



You will see the following window:

Figure 6–3 Add Frameworks and Libraries



See the sections below for more information:

- Section 6.2.3.1, "Frameworks"
- Section 6.2.3.2, "Libraries From The Sync Package"

6.2.3.1 Frameworks

The following frameworks are needed for mobile sync:

- **CoreFoundation.Framework**
- **CFNetwork.Framework**
- **SystemConfiguration.Framework**
- **Security.Framework**
- **UIKit.Framework**
- **Foundation.Framework**
- **Libsqlite3.dylib** - only for Sqlite clients

> **Note:** Berkeley DB clients will use Berkeley DB libraries included in the sync package.

Add these frameworks using the procedure outlined above. You may later move the frameworks under "**Frameworks**" group in your project

6.2.3.2 Libraries From The Sync Package

If you are only building for one architecture (either device only or simulator only), it is simple to add the sync package libraries.

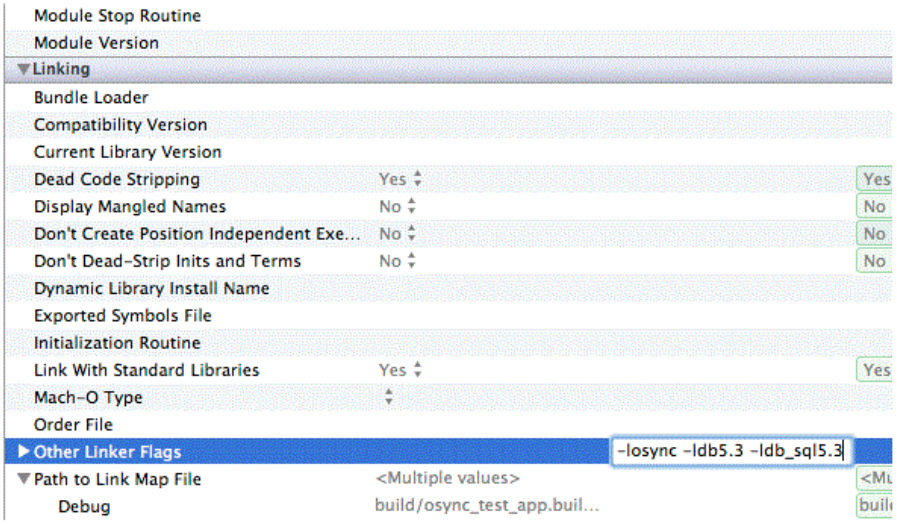
On the window from Figure 6–3, click on "**Add Other**". Navigate to the sync package directory, **lib** subdirectory and then appropriate subdirectories for your platform and CPU type (currently this will only depend on whether you are performing device or simulator build).

You need to add **libosync.a** and **libosync_bdb.a** for Sqlite and Berkeley DB clients respectively. For Berkeley DB clients only, you also need to add **libdb_sql-6.0.a**.

On the other hand, if you want to have both simulator and device architectures you may have to follow different steps:

1. Within the "**Build Settings**" tab (Figure 6–1) navigate to the "**Linking**" section and find "**Other Linker Flags**" row.
2. Click to the right of this row and add linking to these libraries explicitely:
   - **-losync** for sqlite clients
   - **-losync -ldb_sql-6.0** for Berkeley DB clients

Figure 6–4 Explicit Linking



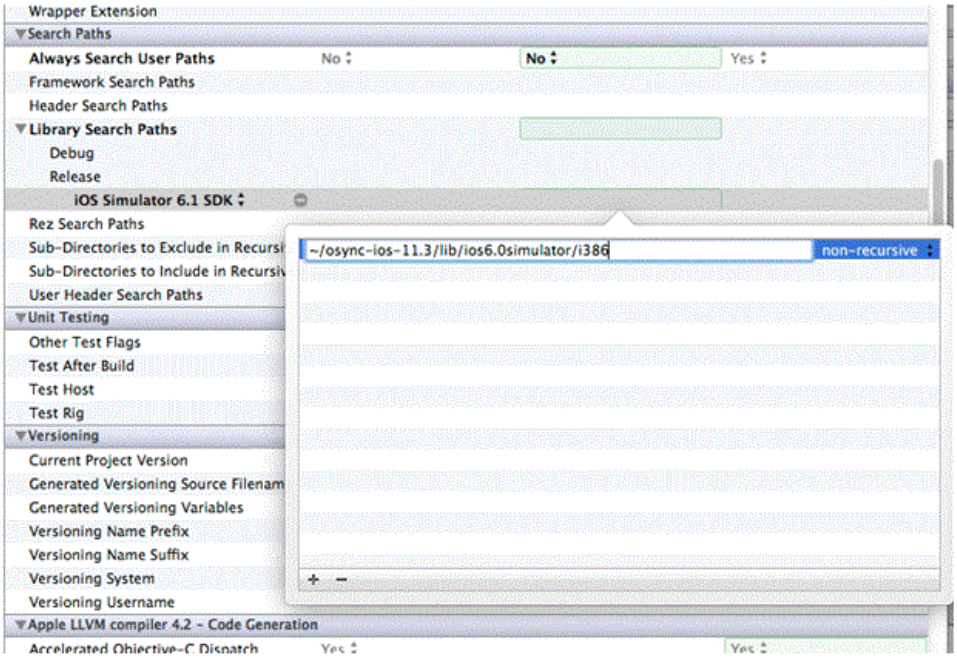
3. The previous step only specified the library names but not their locations. You can customize the search path for different architectures following the steps below:
   - In the same "**Build Settings**" tab, find "**Search Paths**" section and "**Library Search Paths**" row. Toggle it.
   - For each build configuration, "**Debug**" and "**Release**", you can customize the library search path depending on SDK and architecture. Click "**+**" next to configuration. You will see "**Any Architecture | Any SDK**", click on it to pop up the menu.
   - To have both device and simulator builds, it would be enough to customize based on SDK ("**iOS 6.1 SDK**" or "**iOS Simulator 6.1 SDK**").

- Double-click to the right of the selected item. A window will pop-up, click on "**+**" at the bottom left corner.
- For each SDK, enter appropriate path within the **lib** directory of the sync package. See Figure 6–5 below.

Note: You can separately customize Debug and Release configurations.

Now your project will link with correct libraries from the sync package depending on the configuration, architecture and SDK.

Figure 6–5 Customizing Library Search Paths



6.2.4 Include Sync Package Resources

You must include resource from the sync package. Currently they only contain localized string files for error and other messages. More will be added in the future. Follow the steps below to include sync package resource:

1. Right-click on your project and select "**Add Files To [Your Project Name]**". Navigate to "**res**" directory in the sync package, select "**osync_res**" subdirectory and add it to the project.

   Alternatively you can navigate to "**res**" directory in the Finder and drag "**osync_res**" directory to your project.

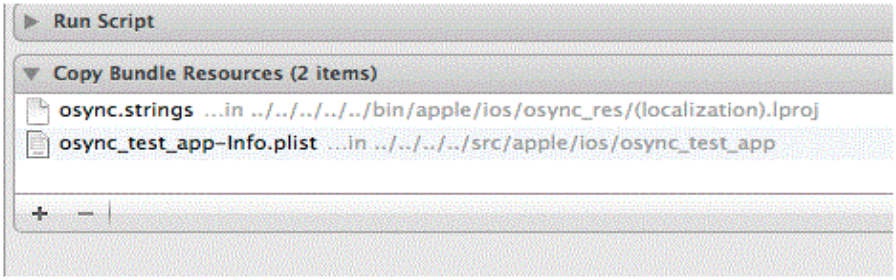
   Note: You can choose "**Copy Files to Destination Group's Folder**" option but it is not required.

   ---

2. Xcode should automatically add the string files within "**osync_res**" to "**Copy Bundle Resources**" build phase. You can verify this by going to the "**Build Phases**"

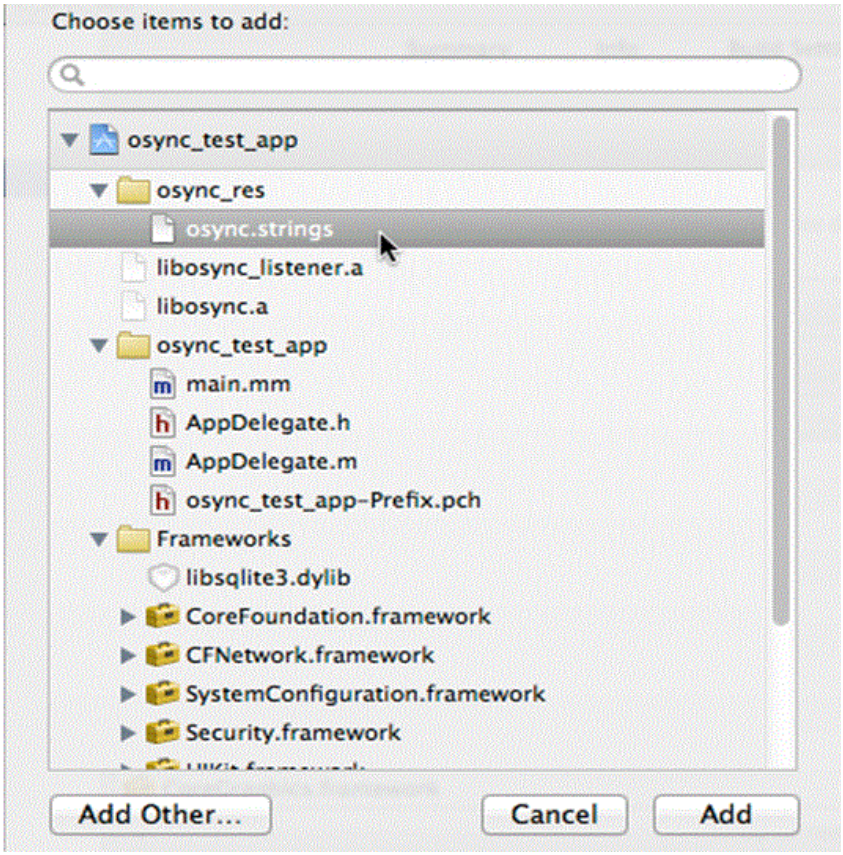
tab and toggling "**Copy Bundle Resources**" section. You should see localized "**osync.strings**" file added.

Figure 6–6 osync.strings in "Copy Bundle Resources" Phase



3. In case you don't see "**osync.strings**" file added, you can add it manually by clicking "**+**" from Figure 6–6. On the screen that pops up, you should see "**osync_res**" folder under your project. Select "**osync.strings**" file in it. See Figure 6–7.

Figure 6–7 Adding "osync.strings" Manually



This concludes the steps needed to build Sqlite and Berkeley DB sync applications.