iOS SimpleSubscribe in 10 minutes with PubNub 3.4 on iOS

This HOWTO will walk you through setting up a simple PubNub 3.4 for iOS "Hello World application". Its a super-simplified walkthrough which utilizes existing code to get you running quickly.

If you'd instead prefer a more robust PubNub iOS example, please refer to the iPad sample app which ships as a demo, available at https://github.com/pubnub/objective-c/tree/master/iOS.

The first step is to clone the PubNub objective-c repo:

```
$ git clone https://github.com/pubnub/objective-c.git
```

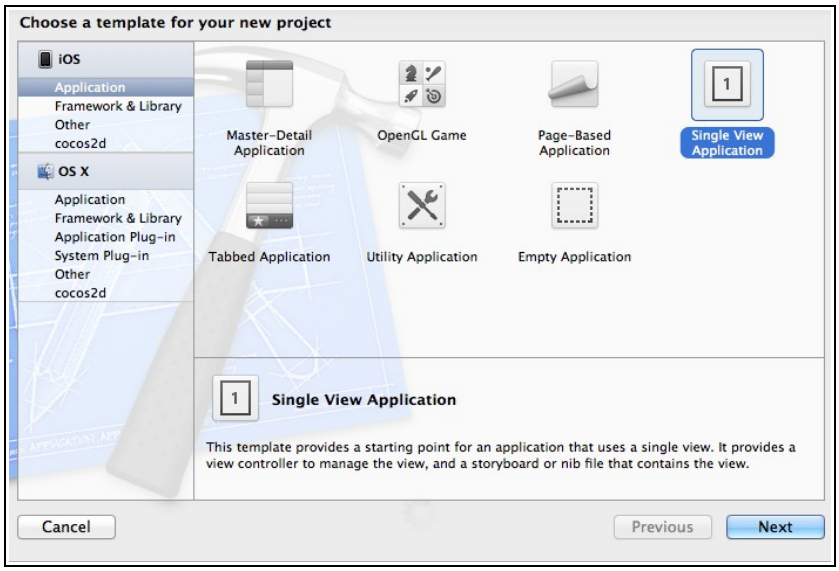
This will create PubNub-related files within a subdirectory called "objective-c" in the same directory you run the command from. You will need these files later.

Create a new PubNubDemo XCode Project

Lets first start with a new blank Xcode project. To create the new project:

1. Open Xcode
2. Select **File -> New -> Project** from the menu.

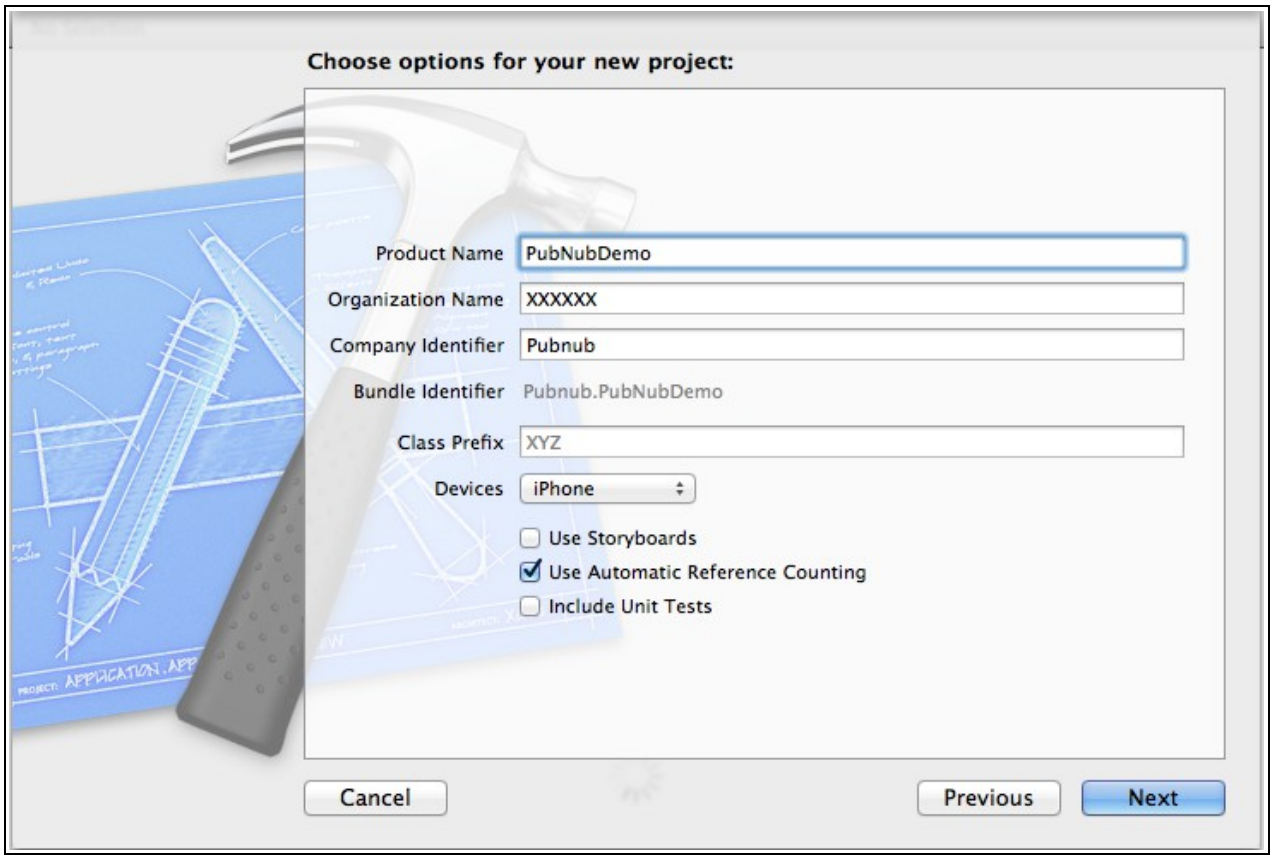
The "**Choose a template for your new project**" dialog will appear.



3. Select **iOS -> Application -> Single View Application**.
4. Click "**Next**".

The "**Choose options for your new project**" dialog will be shown next.

NOTE: In order to make this HOWTO as easy as possible to follow, these following values are suggested:



NOTE: Don't actually enter "**XYZ**" for "Class Prefix". Leave it blank, and it will display "XYZ" as placeholder text.

5. Click the "**Next**" button to proceed.
6. Select the folder which to save the project.
7. Click the "**Create**" button.

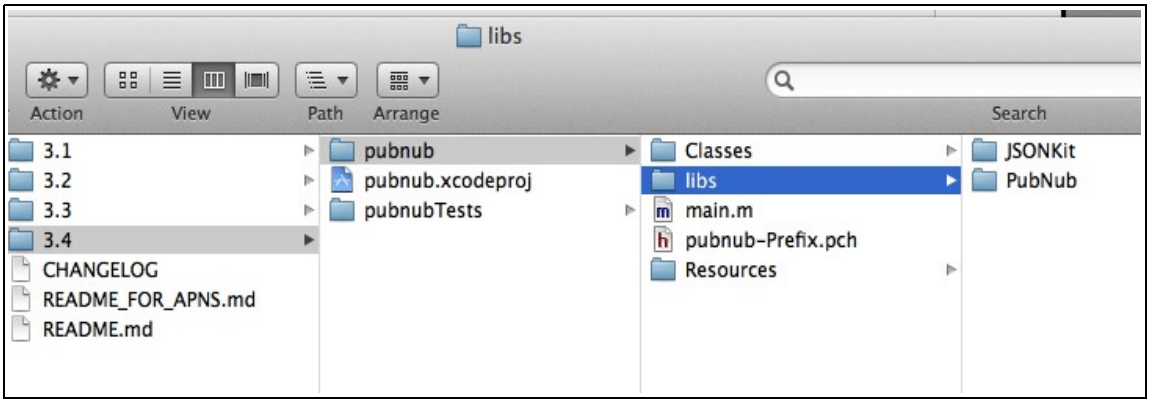
The template project is now created!

Next, we'll import and configure the PubNub libraries we obtained from the git clone performed earlier.

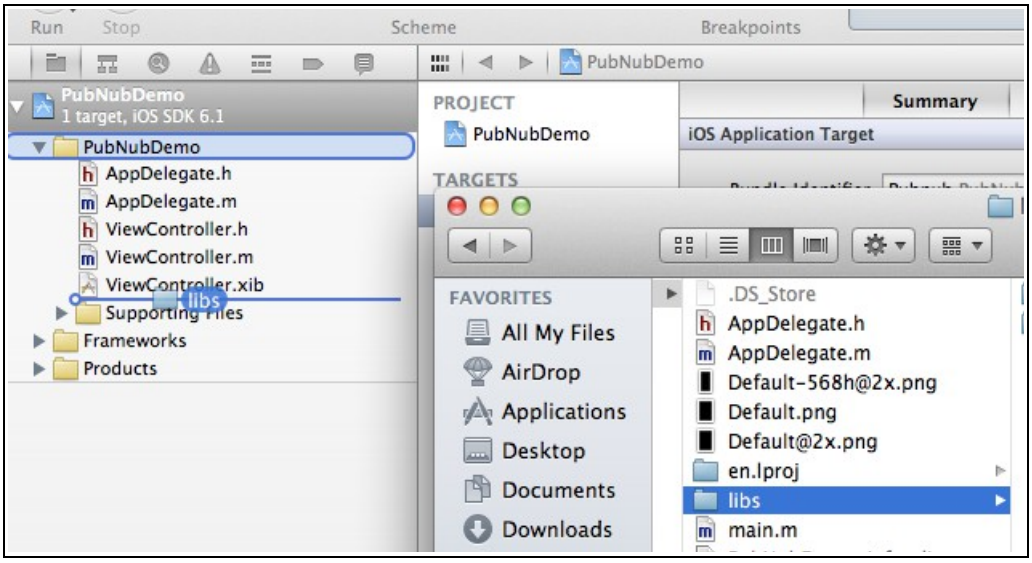
Import and Configure the PubNub Libraries

1. Open a Finder window to the directory created by the previous "git clone", and select the

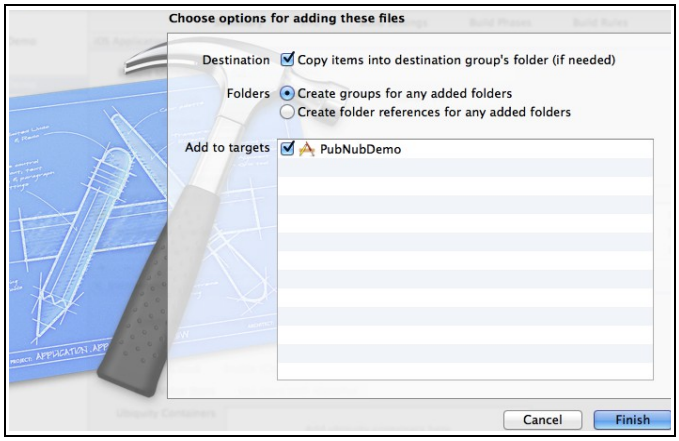
`objective-c/iOS/3.4/pubnub/libs` directory.



2. Drag the **`libs`** directory from the Finder window to just below the file **`ViewController.xib`** in your Xcode project's Project view.



The "**Choose options for adding these files**" dialog will appear.



3. Select "**Destination: Copy items into destination group's folder**" and "**Add to targets**"
4. Click **Finish**.

NOTE: Be sure "**Copy items into destination groups folder**" and "**Add to targets**" is selected for "PubNubDemo" when copying **any** files from the Finder into the project.

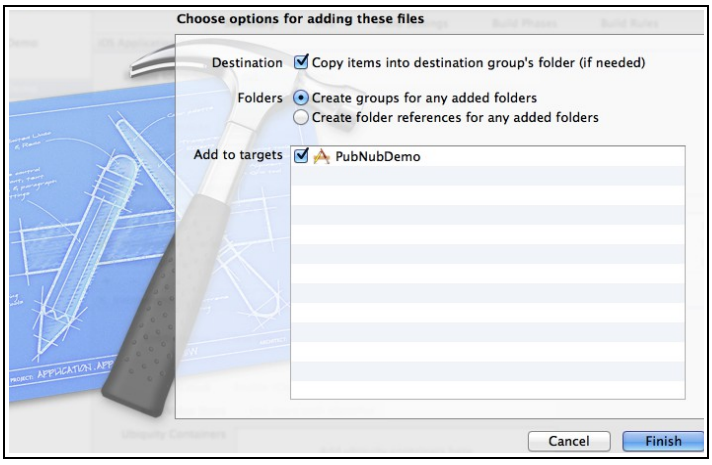
Next, we'll add the "`PNImports.h`" import statement to the **`PubNubDemo-Prefix.pch`** file.

5. Delete **`PubNubDemo-Prefix.pch`** from the path *PubNubDemo/Supporting Files* in Project Navigator.

The "**Move To Trash**" dialog appears.

6. Select "**Move to Trash**"
7. Drag **`objective-c/HOWTO_3.4/PubNubDemo/PubNubDemo-Prefix.pch`** back to *PubNubDemo/Supporting Files* in Project Navigator

The "**Choose options for adding these files**" dialog will appear.



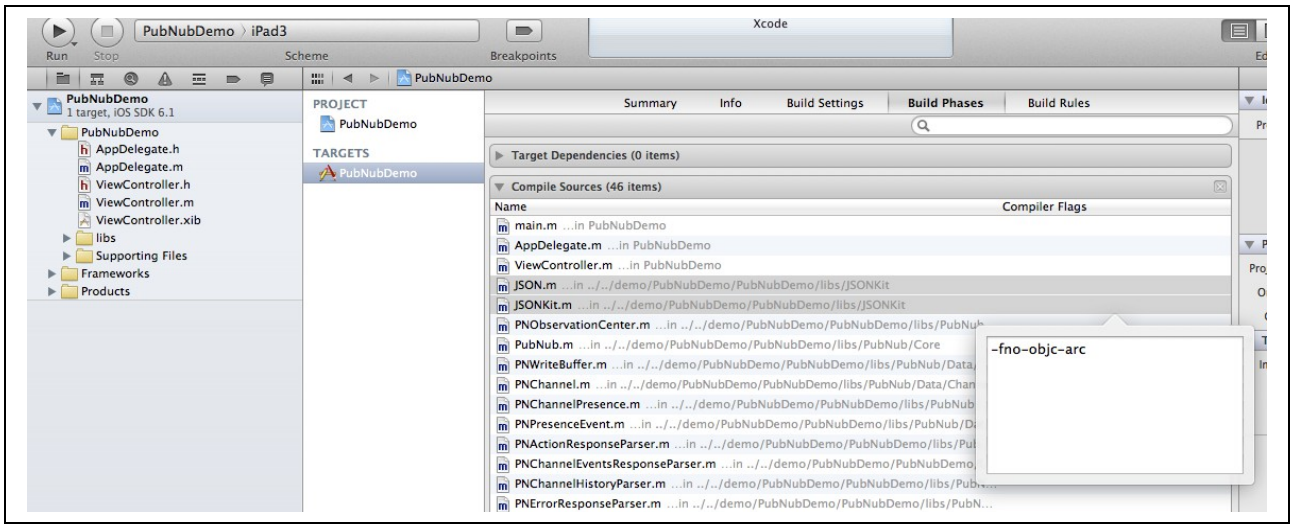
8. Select "**Destination: Copy items into destination group's folder**" and "**Add to targets**"
9. Click **Finish**.

NOTE: Be sure "**Copy items into destination groups folder**" and "**Add to targets**" is selected for "PubNubDemo" when copying files from the Finder into the project.

Disable ARC for Global JSON Support

The JSONKit library is supplied to make supporting all iOS target versions seamless and easy. Since JSONKit library is a non-arc library, we need to tell Xcode to compile these files without ARC.

1. From *Project Navigator* (the 1st vertical Xcode pane), click on the project name **PubNubDemo**
2. In the 2nd pane, click *PubNubDemo* under "**Targets**"
3. In the 3rd pane, from the top horizontal tab menu, click *Build Phases*
4. Expand *Compile Sources*
5. Command-click **`JSON.m`** and **`JSONKit.m`**
6. Press **Enter** to open a flags textfield
7. Paste in the string **`-fno-objc-arc`**



8. Press **Enter** when done.

Next, we need to define the additional framework support required to run our PubNub application. From the current *Build Phases* screen:

9. Close (un-expand) the *Compile Sources* dropdown.
10. Expand *Link Binary With Libraries* dropdown.
11. Click the + button
12. Add **`CFNetwork.Framework`**
13. Add **`libz.dylib`**
14. Add **`SystemConfiguration.Framework`**

Modify the App Delegate files

App delegate logic can handle many PubNub-related tasks, such as receiving messages. We'll implement the didReceiveMessage delegate in this example.

1. Modify your **`AppDelegate.h`** file. Your line 13 should look like this:

```
@interface AppDelegate : UIResponder <UIApplicationDelegate, PNDelegate>
```

2. Modify your **`AppDelegate.m`** file. Immediately after `@implementation AppDelegate at` line 13, add the following delegate method:

```
- (void)pubnubClient:(PubNub *)client didReceiveMessage:(PNMessage *)message {
        PNLog(PNLogGeneralLevel, self, @"PubNub client received message: %@", message);
}
```

3. Modify your **`AppDelegate.m`** file. Immediately before the `return YES;` statement, in the

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
```

method, add the following:

```
[PubNub setDelegate:self];
```

Modify the ViewController files:

1. Modify your **`ViewController.m`** file. Immediately after `[super viewDidLoad];` add the following code to make PubNub connect and subscribe just as the view loads:

```
// amongst other things, set the sub/pub keys to demo
[PubNub setConfiguration:[PNConfiguration defaultConfiguration]];

[PubNub connectWithSuccessBlock:^(NSString *origin) {
   PNLog(PNLogGeneralLevel, self, @"{BLOCK} PubNub client connected to: %@", origin);

   // wait 1 second
   int64_t delayInSeconds = 1.0;
   dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW, delayInSeconds * NSEC_PER_SEC);
   dispatch_after(popTime, dispatch_get_main_queue(), ^(void){

   // then subscribe on channel a
   [PubNub subscribeOnChannel:[PNChannel channelWithName:@"a" shouldObservePresence:YES]];
```

```
    });
 }
    // In case of error you always can pull out error code and identify what happened and what you can do
    // additional information is stored inside error's localizedDescription, localizedFailureReason and
    // localizedRecoverySuggestion)

errorBlock:^(PNError *connectionError) {
    if (connectionError.code == kPNClientConnectionFailedOnInternetFailureError) {

        // wait 1 second
        int64_t delayInSeconds = 1.0;
        dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW, delayInSeconds * NSEC_PER_SEC);
        dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
            PNLog(PNLogGeneralLevel, self, @"Connection will be established as soon as internet connection will be
restored");
            });
        }
        UIAlertView *connectionErrorAlert = [UIAlertView new];
        connectionErrorAlert.title = [NSString stringWithFormat:@"%@(%@)",
            [connectionError localizedDescription],
            NSStringFromClass([self class])];
        connectionErrorAlert.message = [NSString stringWithFormat:@"Reason:\n%@\n\nSuggestion:\n%@",
            [connectionError localizedFailureReason],
            [connectionError localizedRecoverySuggestion]];
        [connectionErrorAlert addButtonWithTitle:@"OK"];

        [connectionErrorAlert show];
}];
```

Completed!

Now, when you run your application, you should see all messages published to the channel in the log. By default (because we are using defaultConfiguration, the subscribe key is set to **demo**, and via the subscribeOnChannel call, listen on channel **a**.