

Add OneSignal APNS to a BT 3 Apple app

(last updated Alanmac, 22 July 2016)

Step 1 - Create an Apple certificate

This is Apple specific process,

Pushwizard instructions <http://support.pushwizard.com/hc/en-us/articles/204849488-Generating-iOS-Certificate>

OneSignal instructions: <https://documentation.onesignal.com/docs/generating-an-ios-push-certificate>

Step 2 - Create a OneSignal Application on onesignal.com

If you don't have a OneSignal account, create one, and then log in to it at <https://onesignal.com/>

Go to <https://onesignal.com/apps/> and create an iOS OneSignal application, platform type Apple iOS (APNS) and click next. You can now upload your Sandbox P12 certificate created in step 1 and a Production Certificate. If you do not have a Production Certificate at this point you can add one later. Click Save, then select your target SDK – Native iOS. Click Next and make a note of the App ID. That's it for now on the OneSignal console, come back to it for testing after the SDK has been integrated into your Apple project in Xcode.

Step 3 - Add OneSignal files to your Xcode Project

OneSignal provide clear instructions for importing the OneSignal framework manually, they have been adapted slightly for a Buzztouch project: <https://documentation.onesignal.com/v2.0/docs/installing-the-onesignal-ios-sdk>

Go to <https://github.com/one-signal/OneSignal-iOS-SDK> and download the iOS SDK zip file from Github.

Unzip the sdk into a newly created folder. Open your project in Xcode and drag OneSignal.framework file into the Xcode FrameWorks folder. Make sure options to 'Copy items if needed' and 'create folder references' and 'Add to targets' are all selected.

Add -ObjC to Target Build Settings>Other Linker Flags

Add SystemConfiguration.framework to Link Binary with Libraries under Target Build Phases.

In Targets, under Capabilities, enable Background Modes and select Remote Notifications.

Step 4 - Make edits to your Project

OneSignal provide clear instructions, they have been adapted slightly for a Buzztouch project:

<https://documentation.onesignal.com/v2.0/docs/installing-the-onesignal-ios-sdk>. All the edits are in your AppDelegate.m and .h file (in the BT_Config folder).

4.1 In the AppDelegate.h file, under the existing import statements (around line 4.1) add the line:

```
#import <OneSignal/OneSignal.h> // add onesignal
```

4.2 In the same file, with the existing @property statements add the line:

```
@property (strong, nonatomic) OneSignal *oneSignal; // add onesignal
```

4.3 In the appDelegate.m file, in the existing didFinishLaunchingWithOptions method, around line 53, add the following lines in yellow, remembering to replace the appID string with the one created in step 2. It should look something like this:

```
//didFinishLaunchingWithOptions...
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
[BT_debugger showIt:self message:[NSString stringWithFormat:@"didFinishLaunchingWithOptions%@", @""]];

/* ajm below added to support onesignal */
self.oneSignal = [[OneSignal alloc]
initWithLaunchOptions:launchOptions
appID:@"xxxxxxx-yyyy-zzzz-abcd-mmmmmmmmmmmmm"]

handleNotification:^(NSString* message, NSDictionary* additionalData, BOOL isActive) {

// This function gets called when a notification is received while the app is in focus.
// code based on the example provided with SDK

AudioServicesPlaySystemSound(1007 ); // play system sound, see http://iphonedevwiki.net/index.php/AudioServices
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate); // vibrate

NSString* messageTitle = @"Alert";
NSString* fullMessage = [message copy];

if (additionalData) {
    NSLog(@"additionalData: %@", additionalData);

    NSString* customKey = additionalData[@"customKey"];
    if (customKey)
        NSLog(@"customKey: %@", customKey);

    if (additionalData[@"inAppTitle"])
        messageTitle = additionalData[@"inAppTitle"];

    if (additionalData[@"actionSelected"])
        fullMessage = [fullMessage stringByAppendingString:[NSString stringWithFormat:@"\nPressed ButtonId:%@",
additionalData[@"actionSelected"]]];
}

UIAlertView* alertView = [[UIAlertView alloc] initWithTitle:messageTitle
message:fullMessage
delegate:self
cancelButtonTitle:@"Close"
otherButtonTitles:nil, nil];

[alertView show]; // show the alert as a pop-up

// Check for and read any custom values you added to the notification
// This done with the "Additional Data" section the dashboard OR setting the 'data' field on our REST API.
};
/* ajm above added to support onesignal */
```

```
/*
Debugging Output
```

Set showDebugInfo to TRUE to print details to the console. To see the console, choose Run > Show Console while the simulator or a connected device is running. Nearly every method has output to show you details about how the program is executing. It looks like lots of data (it is), but it's very useful for understanding how the application is behaving.

```
*/

//show debug in output window?
[self setShowDebugInfo:TRUE];
```

Step 5 - Test

Run a test push from the OneSignal console. OneSignal will only allow a push if the device registers, which normally proves it has been correctly set up. If it has not, check everything again. Common mistakes are the project bundle id does not match the certificate.