



# Appsfire App Booster SDK

v1.1

|  |           |
|--|-----------|
| <b>Requirements</b>  | <b>2</b>  |
| <b>Unzipping the archive</b>                                 | <b>2</b>  |
| <b>Creating your API key</b>                                 | <b>3</b>  |
| <b>It's as easy as 1-2-3</b>                                 | <b>4</b>  |
| <b>Adding your "Call-to-Action"</b>                          | <b>6</b>  |
| <b>Best practice: Navigation Bar or Tab Bar</b>              | <b>6</b>  |
| <b>Alternative Option: The App Booster Notification Bar</b>  | <b>7</b>  |
| <b>Ineffective placement</b>                                 | <b>9</b>  |
| <b>Springboard Badge Count Integration</b>                   | <b>9</b>  |
| <b>Dealing with ARC (Automatic Reference Counting)</b>       | <b>9</b>  |
| <b>Sending substitution data to the SDK</b>                  | <b>12</b> |
| <b>Customizing Colors</b>                                    | <b>12</b> |
| <b>Knowing when the SDK is ready</b>                         | <b>13</b> |
| <b>Knowing when the Notification Wall is being displayed</b> | <b>13</b> |
| <b>SDK Window Styles - Full Screen or Overlay</b>            | <b>14</b> |
| <b>Setting the User's Email</b>                              | <b>15</b> |
| <b>OpenUDID - Alternative to UDID</b>                        | <b>15</b> |
| <b>Advanced - Remote Springboard Badge Updating</b>          | <b>16</b> |
| <b>Managing Notifications</b>                                | <b>17</b> |
| <b>Analytics</b>   | <b>25</b> |
| <b>FAQ</b>   | <b>26</b> |
| <b>Support Considerations</b>                                | <b>26</b> |

## A. Requirements

This document covers the iOS version of Appsfire's AppBooster SDK. If you are looking for the Android version, please visit <http://appsfire.com/appbooster>.

AppBooster SDK runs on versions of iOS 3.2 and above. You will know if the device currently running your app will be able to show the SDK as soon as you initialize it. The SDK can be packaged for all iOS versions; it simply won't do anything on versions of iOS below 3.2. (See instructions for more information)

## B. Unzipping the archive

The archive contains an Xcode integration sample.

Unzipping the archive exposes the project so that it can be compiled and tested.

### Running and testing sample project

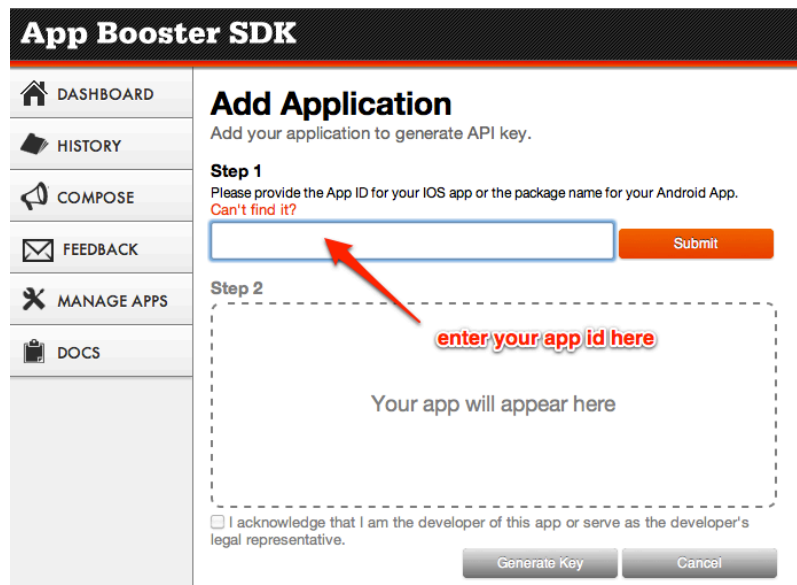
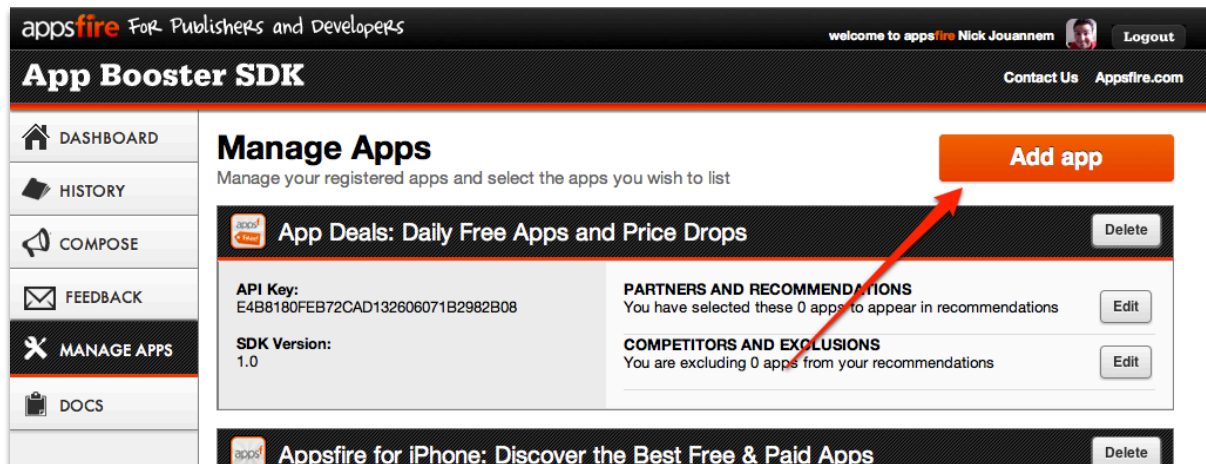
Now, double-click on the AFAppBoosterSDK.xcodeproj sample project. Please make sure that the project compiles and runs.

- ▶ *Note that you will have to enter your own Appsfire API key inside the API call within the application delegate method `appDidFinishLaunching`. Jump to Section C to see how to generate your own private key.*

## C. Creating your API key

Please go to <http://appsfire.com/appbooster> and log in. The easiest way is to use the Facebook login button if you don't have an account.

Register your application by providing your app id or package name as follows:



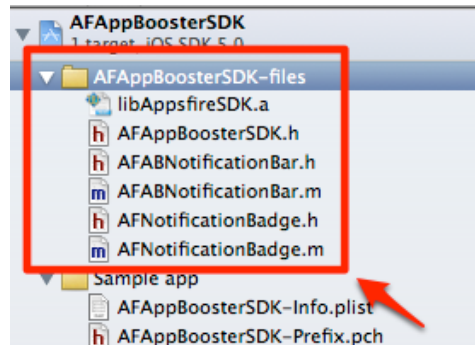
By pressing the Generate Key, you will be provided with a unique API key that you need to paste into the line of code mentioned above in Section B - Step 3, as follows:

```
[AFAppBoosterSDK connectWithAPIKey:@"INSERT YOUR KEY"];
```

## D. It's as easy as 1-2-3

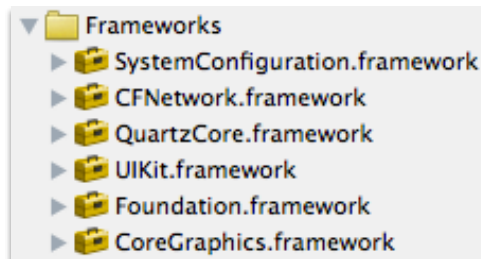
### Step 1. Dropping the assets inside your project

The sample project displays the simplest way to install the SDK inside your app, by drag & dropping the folder “AFAppBoosterSDK” into your own project from the sample project. If prompted, copy the items into your folder.



### Step 2. Making sure all frameworks are included

The Appsfire SDK requires the following Frameworks to operate. Make sure to add them to your project.



## Step 3. Enter the 1 line of code to initialize the SDK

To start, make sure you import the header file called "AFAppBoosterSDK.h".

```
#import "AFAppBoosterSDK.h"
```

Then, inside the didFinishLaunching call-back method of your application delegate:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions
{
    NSString *apiKey = [NSString stringWithString:@"INSERT KEY HERE"];
    [AFAppBoosterSDK connectWithAPIKey:apiKey];
}
```

Once you've set your API key, you may present the SDK however fits your needs by simply calling the following piece of code:

```
[AFAppBoosterSDK presentNotifications];
```

For example, you could use this from a button placed inside your interface. At this point, your project should build and run, and display the SDK Notification Wall if you have created a valid SDK key.

### Advanced

If your app has a lot of things to load from the web at startup, you may want to delay the initialization of the AppBooster SDK by a few seconds to give your app full bandwidth. You can do this by using the following method (also used in the demo app) :

```
NSString *apiKey = [NSString stringWithString:@"INSERT KEY
HERE"];
if ([AFAppBoosterSDK connectWithAPIKey:apiKey afterDelay:2.0])
    NSLog(@"Appsfire Appbooster Demo App launched with %@",
[AFAppBoosterSDK getAFSDKVersionInfo]);
else
    NSLog(@"Unable to launch Appsfire Appbooster Demo App.
Probably incompatible iOS.");
```

In the above example, the delay is 2 seconds. It is your responsibility to prevent the user from trying to launch the SDK window prior to the delay period. If a call to present the notifications wall is made, an error message will appear to the user inviting him/her to try again later.

# Integrating App Booster in your app

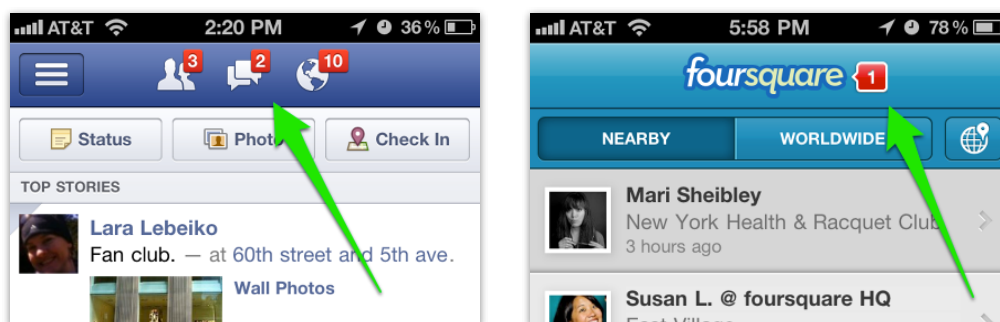
## A. Adding your “Call-to-Action”

### I. Best practice: Navigation Bar or Tab Bar

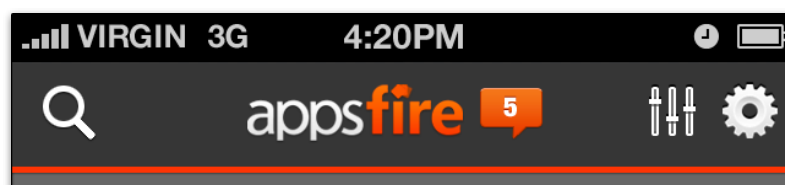
Before we elaborate on best practices, please note that as part of the App Booster SDK service, we are happy to assist you in determining the best placement. Our product designer ([jonathan@appsfire.com](mailto:jonathan@appsfire.com)) is available by email or Skype to consult with you. We are even able to supply you with icons and/or buttons that will suit your needs. Please feel free to reach out.

Through experience in our apps, measurement of the alpha-stage App Booster adopters, and observation of the latest trends in mobile UI/UX, it's clear that the best place to place an engagement-based call-to-action in your app is in the navigation bar. The objective is to give users a prominent and reliable place to go to get the latest notifications from your app so that as soon as they open the app, *they know just where to go*. And for uninitiated users, the call-to-action will be *impossible to miss*.

Within the navigation bar, there are two positions that best convert users and increase engagement/intention. Following the lead of **Foursquare** and **Facebook**, the center of the navigation bar, whether dead-center or offset from your logo, converts best.



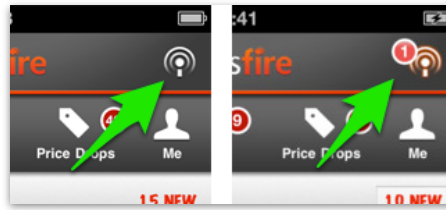
Facebook & Foursquare exhibit the best practice in placement of their call-to-action



In the upcoming version of Appsfire, we've adopted the model innovated by Facebook & Foursquare

If the center of the navigation bar doesn't work for your app, the right-hand side of the navigation bar is also an effective placement. In the current version of Appsfire, we placed an

icon in the right-hand side of the navigation bar that took on a bright orange glow when there was at least one notification waiting.



**A “glowing” call-to-action in the right-hand side of the navigation bar**

Another popular and effective placement for the call-to-action is the nav bar. Similar to placement in the nav bar, this provides a persistent, reliable and prominent placement for access to the Notification Wall.



**A Notifications tab in your tab bar**

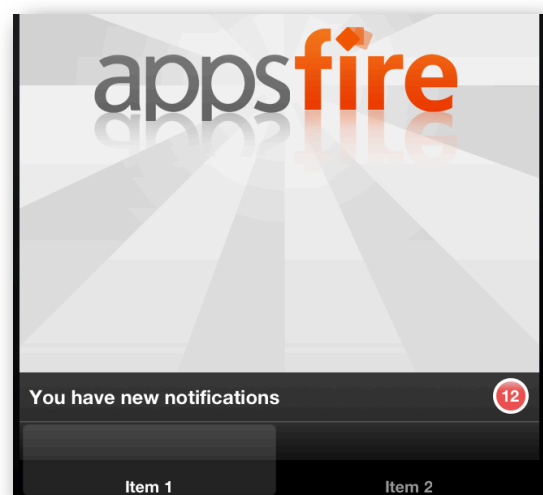
## **II. Alternative Option: The App Booster Notification Bar**

The App Booster library provides an out-of-the-box solution if the navigation bar is not the right placement for your call-to-action.

The notification bar is a subtle bar that will appear on the bottom of the user's device or just above toolbars and tab bars and will provide your users with quick and easy access to the Notification Wall. The text of the bar is automatically updated and translated based on the user's preferences and the number of unread notifications appears in the badge on the right.

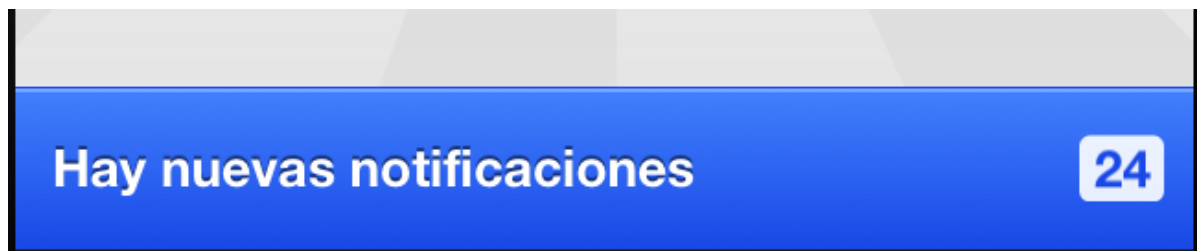


**Example 1 : Simple View Controller**



**Example 2 : Tab Bar Controller**

If you do implement the out-of-the-box Notification Bar, note that modifying the look-and-feel is very simple. In the example below, we've added a glossy gradient. You may opt to set a gradient for this bar that matches your app's color scheme.



**A customized Notification Bar**

To add the notification bar, import AFABNotificationBar.h. Then, add the following lines to the viewDidLoad function of your app's parent view controller:

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    /*
        TO ADD THE NOTIFICATION BAR, include the header file and allocate
        it here. By default, the bar will position on the lower part of the current
        view's frame. You can override this by changing the frame after allocating
        or modifying the code directly.
    */

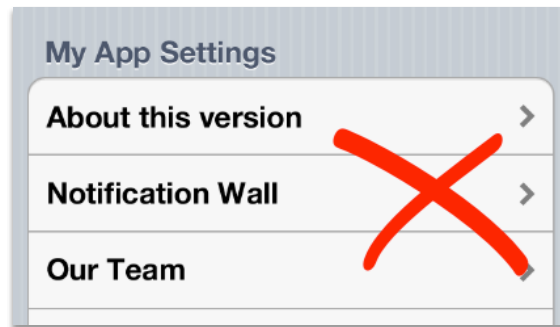
    AFABNotificationBar *bar = [[AFABNotificationBar alloc]
initWithFrame:self.view.bounds];
    [bar setShowWhenCountIsZero:YES];
    [self.view addSubview:bar];
    [bar release];
}
```

*NOTE : setShowWhenCountIsZero is set to NO by default and controls whether or not the bar should be visible when there aren't any new and unread notifications. Setting this to NO will make the notification bar disappear if all notifications have been read.*



### III. Ineffective placement

Two of our alpha-stage developers initially opted to implement in subviews of their app (e.g., one developer placed it in their Settings view) and the results were predictably underwhelming. Upon moving their call-to-action to the navigation bar, both developers experienced exponential improvements in engagement.



How not to implement the call-to-action to the Notification Wall

Even if the Navigation Bar or out-of-the-box Notification Bar do not suit your needs, feel free to take the supplied Notification Bar code apart to leverage the bubble and callback mechanism and find an effective solution.

Once again, if you are struggling to find placement for your call-to-action, we do encourage you to reach out to our product designer ([jonathan@appsfire.com](mailto:jonathan@appsfire.com)) who will readily assist you with finding an elegant solution that fits your app's design/experience.

### B. Springboard Badge Count Integration

Optionally, you can have the badge number of your application appearing on your user's springboard reflect the number of pending notifications the user has. By doing so, if your user exits your app, the number of unread messages will be put on your app's icon, reminding your user that he or she should come back and visit your app. To enable this feature, you must call the following line of code :

```
[AFAppBoosterSDK handleBadgeCountLocally:YES];
```

See "[G. ADVANCED - Remote Springboard Badge Updating](#)" for how this can be done when your app is closed.

### C. Dealing with ARC (Automatic Reference Counting)

If you come across ARC warnings and errors because of files that aren't ARC-compatible, you can specifically disable ARC on these files by going to your project settings, selecting your app in the Targets, going to the tab "Build Phases", and adding the argument "-fno-objc-arc" to all the files you want ARC to ignore.

## Notification Bar Language Display Setting

The Notification Bar provided ships with English text only. You'll notice that the `AFNotificationBar` class comes with a few variants of its constructor which allow you to also specify a `displaySetting`. There are 3 constants that can be used when passing a `displaySetting`. (Note: A detailed description is also found in the header file). The available settings are :

### `kAFABNotificationBarSetting_DisplayNowForceToEnglish`

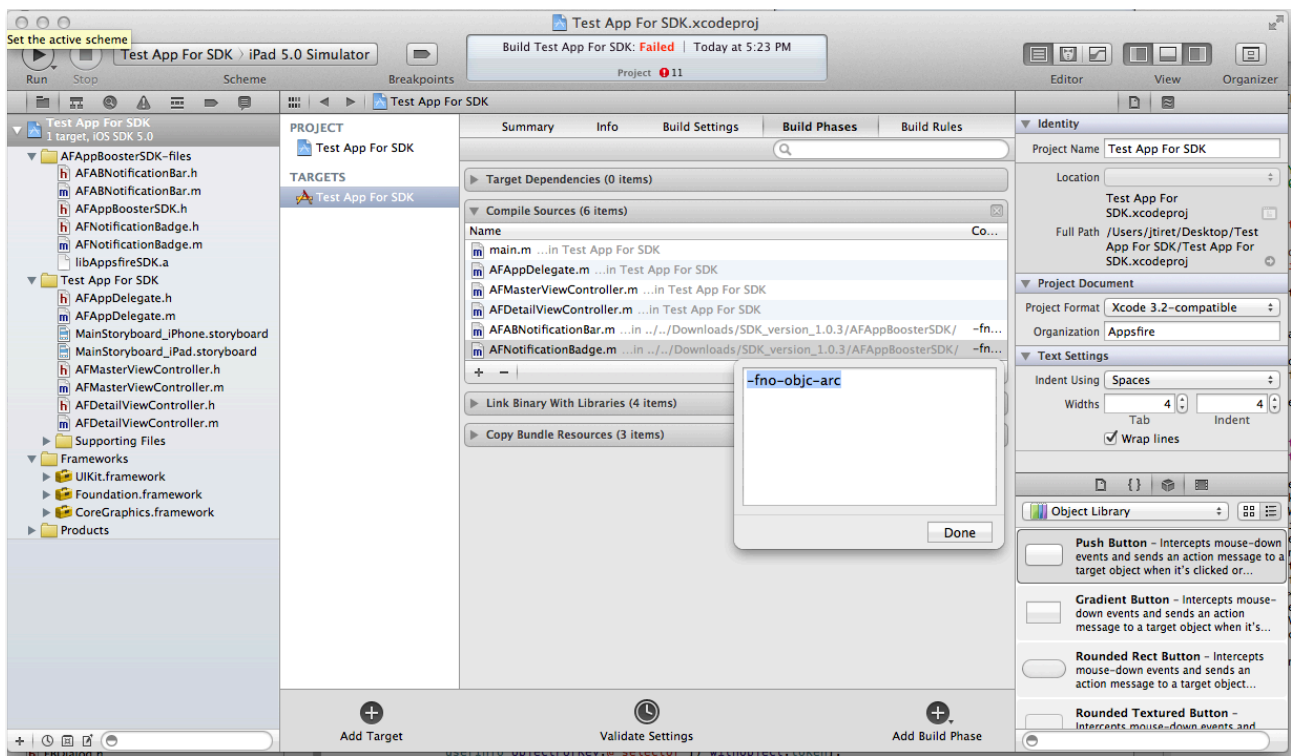
This constant will force the use of English strings during the user's first session. Even if the strings are updated later on automatically via the SDK, the text on the bar will remain in English. The next time the user runs your app however, localized strings will be used in place if they are available.

### `kAFABNotificationBarSetting_DisplayNowInDefaultLanguageAndRefreshToLocaleStringsAsSoonAsTheyBecomeAvailable`

This is the default and recommended setting. The Notification Bar will load as soon as it's ready (almost instantly) using English strings. As soon as the SDK has detected your user's locale and downloaded localized versions of the strings, the text on the bar will change to the user's locale language. As a result, your user may briefly see English text for a moment before it automatically switches to his or her local language.

### `kAFABNotificationBarSetting_DisplayOnlyWhenLocaleStringsAreReady`

This setting will hide the Notification Bar, even if there are notifications to show, until the SDK has taken the necessary time required to check and display the strings in the user's local language. Given the dependence on an outside source of data, this method is not recommended but is made available to you should you require this behavior.



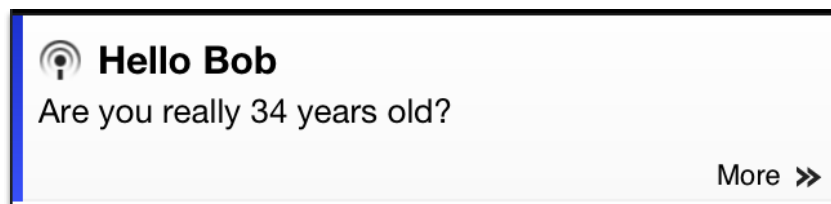
## D. Sending substitution data to the SDK

In some cases, you may want to supply the SDK with data. You can do so by sending an NSDictionary of key/value pairs that will then be substituted whenever possible.

Example: You create a notification in the SDK back-office. You want the first line of text to say “Hello Bob, you’re 34 years old” where “Bob” is your user’s first-name and 34 is his age; you’ve collected this information through your own processes. You could do this :

```
NSDictionary *customTags = [NSDictionary
                           dictionaryWithObjects:[NSArray
arrayWithObjects:@"Bob",@"34",nil]
                           forKeys:[NSArray
arrayWithObjects:@"FIRSTNAME",@"AGE",nil]];
[AFAppBoosterSDK useCustomValues:customTags];
```

By doing this, any instance of the string [FIRSTNAME] or [AGE] will be replaced by the corresponding values. Here’s an example :



## E. Customizing Colors

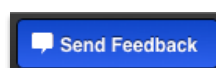
If you would like to customize the standard orange colors of the SDK, you may pass 2 colors that will be used to create a gradient (linear top to bottom). For example :

```
[AFAppBoosterSDK useGradients:[NSMutableArray arrayWithObjects:
    [UIColor redColor], [UIColor greenColor], nil]];
```

or

```
[AFAppBoosterSDK useGradients:[NSArray arrayWithObjects:
    [UIColor colorWithRed:.1137 green:.227 blue:.776 alpha:1.0],
    [UIColor colorWithRed:.2196 green:.341 blue:.933 alpha:1.0],
    nil]];
```

The latter would produce a Feedback button with the following appearance :



## F. Knowing when the SDK is ready

When the SDK is ready to be presented, it will emit the event AFSDKIsInitialized. You can listen to the event like so:

```
[[NSNotificationCenter defaultCenter] addObserver:self  
selector:@selector(sdkIsInitialized) name:@"AFSDKIsInitialized" object:nil];
```

where sdkIsInitialized is your own function. The AFABNotificationBar.m file in the sample project provides an example of this.

You can also query the SDK's state using :

```
[AFAppBoosterSDK isInitialized];
```

which will return a Boolean value.

## G. Knowing when the Notification Wall is being displayed

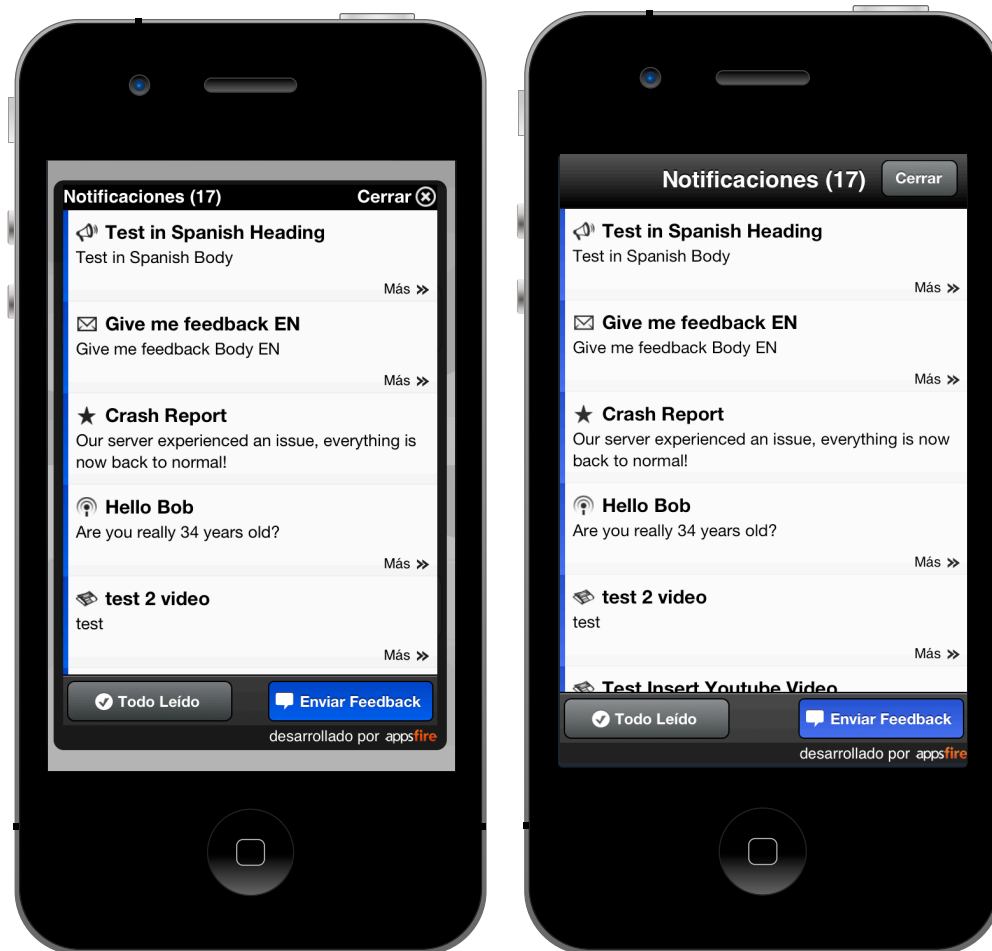
In some cases, you may need to know when the Notification Wall is actually being displayed. You can get this information simply:

```
[AFAppBoosterSDK isDisplayed];
```

which will return a Boolean value.

## H. SDK Window Styles - Full Screen or Overlay

The default style of the SDK is the Overlay (left). However, you may opt for the full screen version (right). These versions are identical feature-wise.



To use the full screen version, call :

```
[AFAppBoosterSDK useFullScreenStyle];
```

## E. Setting the User's Email

If you already know your user's email address, you can improve the user experience of sending feedback by telling it to Appbooster. By doing so, the feedback form will have your user's email pre-filled.

```
[AFAppBoosterSDK setUserEmail:@"user@email.com" isModifiable:YES];
```

The "isModifiable" parameter will make the email field of the feedback form visible or not. Even if not modifiable, you can set the email so that it appears on the email you receive.

## F. OpenUDID - Alternative to UDID

As of June 2011, Apple has indicated that it will deprecate the method `[[UIDevice currentDevice] uniqueIdentifier]` that allows developers to access a unique identifier per device. Since device tracking is essential in allowing App Booster to deliver the right messages to the right user without requiring any personal information, it comes built-in with OpenUDID. OpenUDID is an open-source UDID alternative that allows apps to track users with a unique identifier while preserving the user's privacy and at all times and across multiple apps using OpenUDID.

If you wish to use and access your user's OpenUDID, simply call :

```
[AFAppBoosterSDK openUDID]
```

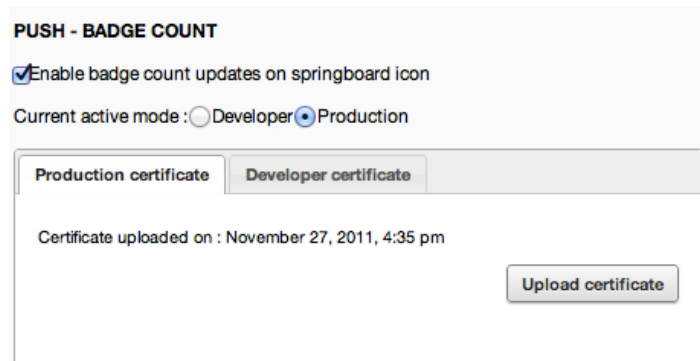
In the event where the user has opted out of OpenUDID, a zero-filled string of 40 characters long will be returned.

This call is available as of version 1.0.5 of the Appbooster SDK.

## G. Advanced - Remote Springboard Badge Updating

When handling badge display and updating, you have the option of enabling your app to have its springboard badge updated even when your app is not active on the user's device. In order for this to happen, you must :

- Use `[AFAppBoosterSDK handleBadgeCountLocallyAndRemotely:YES];`
- Enable badge count in the back-office
- Upload a Production Push Certificate and set it as the Active Mode
- Add push token handling code to your app.



Client side, you must register for push notifications using the standard code provided by Apple. This code should be run everytime your app is run. Below is an example :

```
[[UIApplication sharedApplication]
 registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge |
                                     UIRemoteNotificationTypeSound |
                                     UIRemoteNotificationTypeAlert)];
[AFAppBoosterSDK handleBadgeCountLocally:YES];
```

Finally, you must create both callback functions to register your user's push token with Appbooster SDK :

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    NSString *token = [[[deviceToken description]
                        stringByReplacingOccurrencesOfString:@"<" withString:@""]
                        stringByReplacingOccurrencesOfString:@">" withString:@""]
                        stringByReplacingOccurrencesOfString:@" " withString:@""];
    [AFAppBoosterSDK registerPushToken:deviceToken];
    [AFAppBoosterSDK handleBadgeCountLocallyAndRemotely:YES];
}

- (void)application:(UIApplication *)application
didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
{
    [AFAppBoosterSDK handleBadgeCountLocally:YES];
}
```

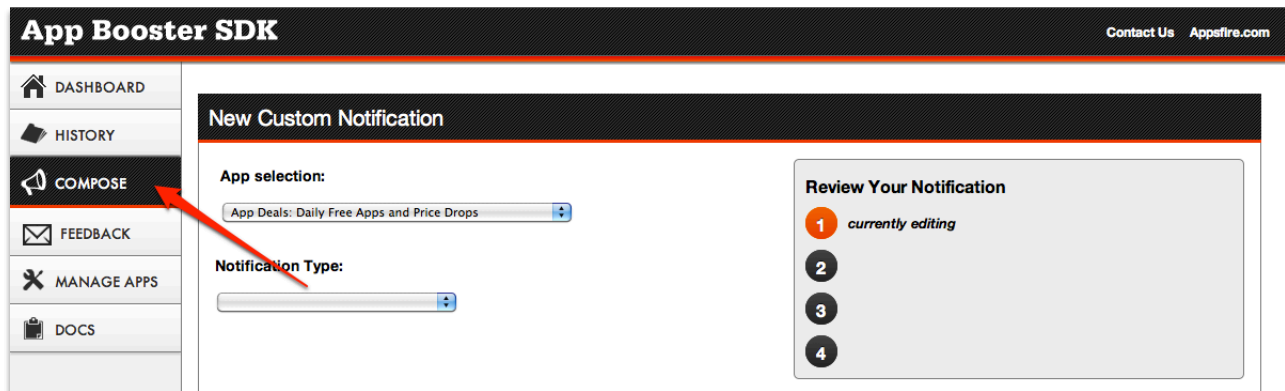
We strongly recommend you look at the demo app to see a full integration of this method.



## H. Managing Notifications

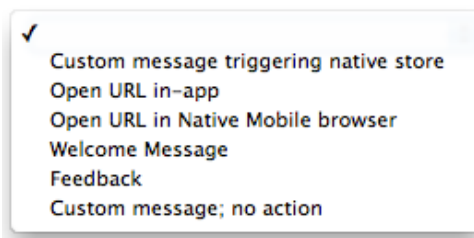
To manage notifications, you will have to log into the SDK section of the Appsfire website, currently located at <http://appsfire.com/appbooster>.

Currently, you may send a notification through the SDK within your app by selecting “Compose” in the navigation bar (see image below). Once here, you will have a series of chronological steps to follow in order to create your notification.



### Step 1 - Choose your Notification Type

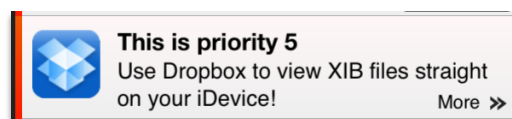
#### Notification Type



There are currently 6 notification types. Each notification type determines the look and feel of your notification as well as what kind of action it will perform.

#### Custom message triggering native store

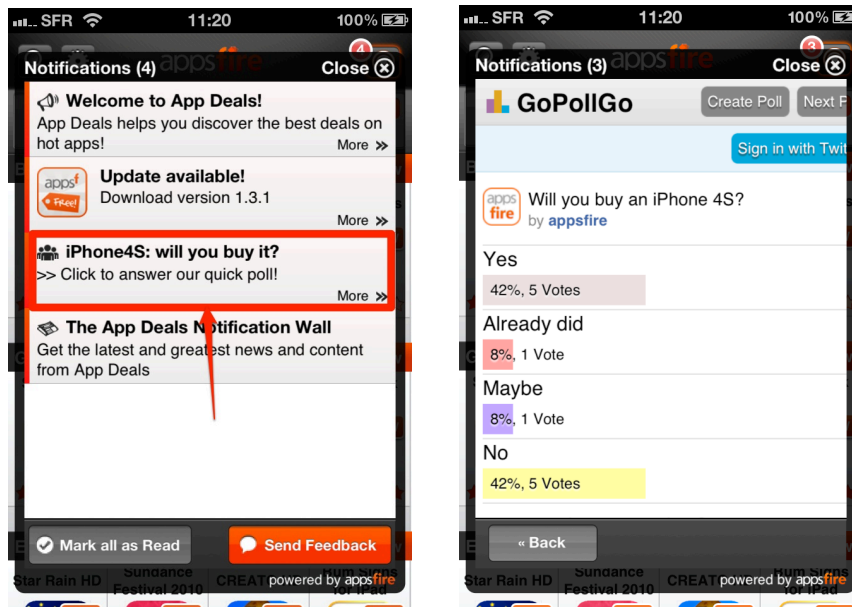
This type of notification is to be used when your notification should open the native app store of your user's device in order to land on a specific app's download page. You will have to provide the app ID or package name when creating this type of notification. Once you've done so, the app icon of the app will automatically be added to the notification.



#### Open URL in-app

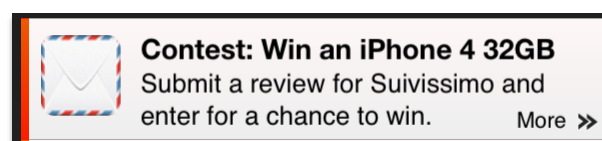
This type of notification will allow you to provide a URL of your choice. When the user clicks on this notification, the URL you provided will open inside the SDK, thus allowing your user to view an external page without ever needing to leave your app.

For example, let's imagine you want to collect survey data from your users. Create a poll with Google Forms, Poll Daddy or as illustrated here, GoPollGo, and get results within minutes!



## Open URL in Native Mobile browser

This type of notification will allow you to provide a URL of your choice. When the user clicks on this notification, the URL you provided will open inside the native mobile browser of the user's device.



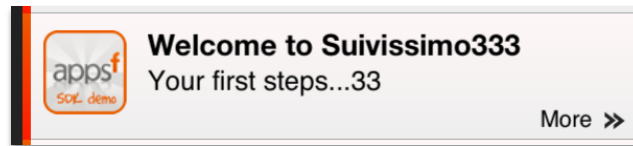
Note that the user will be prompted to leave the app.

### Advanced usage

Properly formed URL handlers will also work. For instance, if your target URL is fb://, the notification will attempt to launch the Facebook app if available (and fail gracefully otherwise). Make sure that the app is installed to avoid frustrating your users.

## Welcome Message

We strongly encourage you to provide a welcome message for the first time your users open your app and trigger the SDK. Only one welcome message will be displayed (the most recent one), so don't hesitate to update it or create a new one as much as you'd like. You can have translated version of your welcome message to better target your users.



In Step 3 (detailed further below) you will be able to provide the HTML that makes up the content of your welcome message. Be creative! When the user clicks on this cell, a welcome notice will slide up with your content.







## Feedback

If you want to get some feedback from your users about a specific issue or just general feedback, you can send them a feedback notification which will trigger the feedback form found inside the SDK.

The image shows two parts of the feedback system. The top part is a screenshot of a mobile app interface showing a 'Notifications (0)' banner with a 'Close' button. Below the banner is a 'Like the app?' section with thumbs up/down icons, and a 'Feedback Type' section with 'Bravo', 'Bug', and 'Idea' buttons. Below that is an 'Email' input field. The bottom part is a desktop view of the 'Notification Type' configuration form. It has a dropdown menu set to 'Open URL in-app', a 'URL:' input field, an 'Icon (optional):' section with various icons (signal, calendar, gift, people, heart, envelope, megaphone, share, folder, laptop, star, signal, globe, globe) and a 'clear' button, and a 'Next >>' button at the bottom.

Feedback will be collected and rendered inside the dedicated "Feedback" section.

Note that each feedback message may be responded to directly by email (if the email has been provided by the user in the form), or via the SDK back office.

| ALL FEEDBACK IN THE PAST WEEK  |   |   |
|--|---|---|
| All Feedback   |   |   |
|  France<br>Nice |  Bravo | October 13, 2011 at 02:12<br><a href="#">Reply via SDK</a>                    |
|  France         |        | October 12, 2011 at 23:05<br><a href="#">Reply via SDK</a>                    |
|  France<br>test |        | October 12, 2011 at 10:18<br><a href="#">Reply via SDK - Reply via E-Mail</a> |

## Custom message; no action

This notification will simply display text in the notification window with no action attached to it. It can be marked as read by the user.

### General notes

Depending on the type of notification you choose, you will be asked to provide additional information and optionally choose an icon:

## Step 2 - Choose your Audience

**Geography**

**Available**

- Afghanistan
- Aland Islands
- Albania
- Algeria
- American Samoa
- Andorra
- Angola
- Anguilla
- Antarctica
- Antigua and Barbuda
- Argentina

**Targeted**

>

<

Select All

Remove All

**Publication date**

**from** (optional)

00 : 00

**To** (optional)

00 : 00

**Target**

**Minimum OS Version**

All OS versions

**App Version**

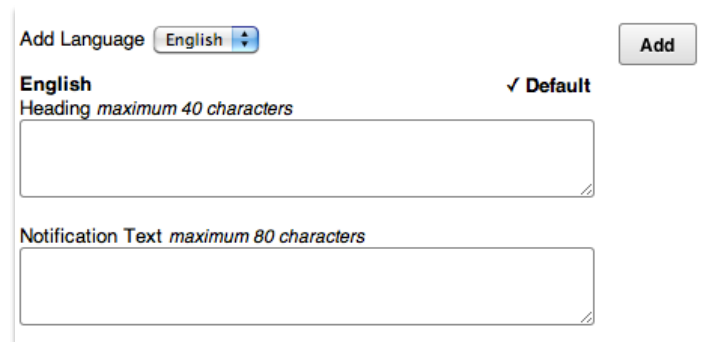
All versions

In Step 2, you will be asked to choose which geographical regions you wish to target, when you wish to release the notification, when you wish to expire the notification, what minimum iOS version you are targeting, and which minimum app version you are targeting. All fields are optional except for the geography selection. If you do not specify any of the other fields, an immediate release date with no expiration will be assumed.

Setting an expiration date will prevent notifications from being sent after this date and will remove notifications from your users notification list once this date is reached.

► *Note: The geography is linked to the user's IP address.*

### Step 3 - Set your text



Add Language English Add

**English** ✓ Default

Heading maximum 40 characters

Notification Text maximum 80 characters

In this step, you will provide the actual content of the notification in as many languages as you'd like.

You can select a default language for all users whose device language does not match any of the languages you set for the notification. For example, if you create a notification in English and Spanish with English as the default, then users with Japanese-language devices will get the notification in English because it's set as the default.

To add multiple languages, simply select the language you'd like to add and click Add.

- ▶ *Note: Your entries will try to match the user's default language as configured on his/her device, not based on geography. Typically, Spanish will reach users in Spain and Mexico, but default to English if Spanish is not provided.*

## Finally - Review, (Test), Create

**Test**  
How to setup

Register New Test Device Test this Notification

Create Notification

**Review Your Notification**

|   |  |                                    |                      |
|---|--|------------------------------------|----------------------|
| 1 | <b>Notification Type</b><br>Feedback                             | <b>Target</b><br>n/a               | <a href="#">Edit</a> |
| 2 | <b>Valid from</b>  | <b>Valid to</b>                    | <a href="#">Edit</a> |
| 3 | <b>OS Verison</b><br>All OS versions<br><b>Geographies</b><br>AD | <b>App Version</b><br>All versions | <a href="#">Edit</a> |
| 4 | <b>Done</b>  |                                    |                      |
|   | <b>currently editing</b>   |                                    |                      |

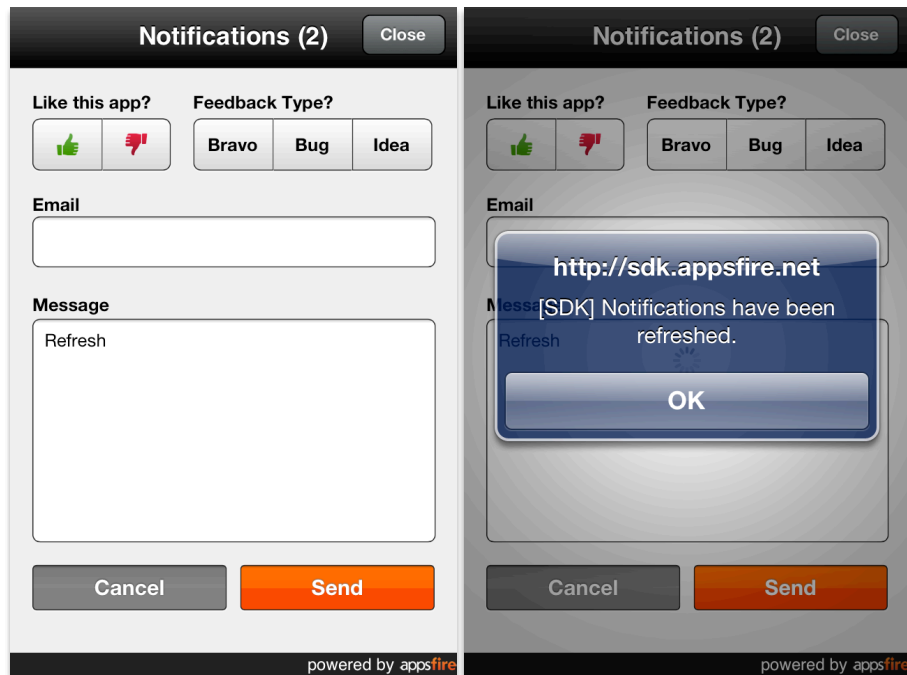
The last step allows you to review your settings and create the notification. If you wish to see what your notification will look like before-hand, you can test it.

In order to test a notification, you must have the SDK running on a device (or simulator) and you must register the device as a “testing device”. Follow the instructions that are displayed when you click on “Register New Test Device” to do this.

Once your device is registered, you can click on “Test this Notification”. This will make the notification available to all registered test devices linked to your SDK.

- ▶ *Note that test notifications behave differently from normal notifications. They cannot be “Read”, even by the “Mark all as read” button. Please be aware that your notification badge count may also not meet your expectations.*
- ▶ *Warning: The SDK has a cache mechanism. If you recently opened the SDK from within your testing environment or device, the cache may prevent you from downloading the test notification. To bypass this, you can wait and try again later or you can erase the installed version of the app hosting the SDK you have and re-install it - this will force the SDK to re-download all of your notifications, including the test ones.*

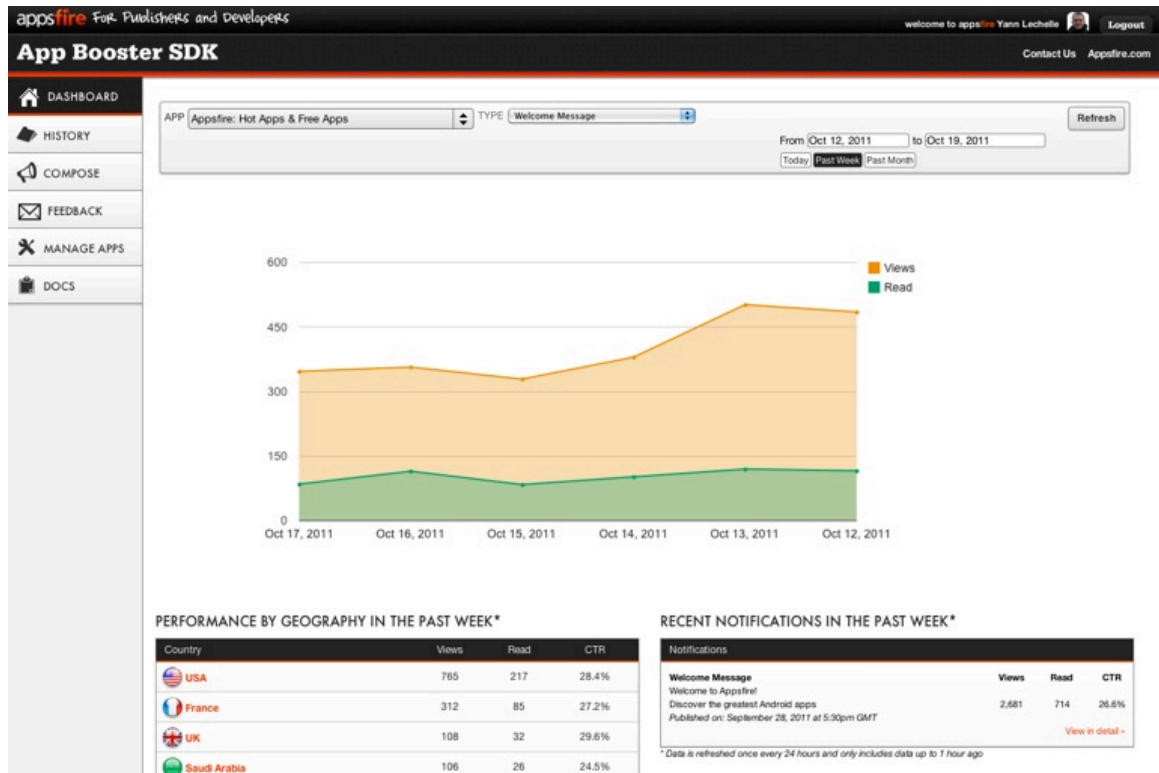
Since version 1.0.2, you can force a refresh of notifications by opening the Feedback Window and typing "Refresh". If your device is successfully registered as a test device, a popup will appear to confirm the refresh was taken into account:





# I. Analytics

Once you have released a few messages, check the dashboard for in-depth analytics, filtered by app, notification type, and date.



Or visit the History tab, and drill down to see how each of your notifications did.

## J. FAQ

Please send all your early comments to [nick@appsfire.com](mailto:nick@appsfire.com).

We are in the process of consolidating the FAQ with our early SDK partners.

### **How much of a size increase should I expect?**

Your app should not increase by more than 250KB after you compile it.

### **I keep getting an Internet Connection error**

This could be caused by one of several reasons :

- 1) You haven't set an API key
- 2) The Notification Wall is being triggered while not being initialized
- 3) You don't have an internet connection

## K. Support Considerations

Please contact [support@appsfire.com](mailto:support@appsfire.com) for general inquiries.

High-level inquiries should be directed to both [yann@appsfire.com](mailto:yann@appsfire.com) and [ouriel@appsfire.com](mailto:ouriel@appsfire.com)