App Booster Advertising SDK

v0.5.4 (Updated on April 17th, 2013)

A. Requirements

Advertising SDK runs on versions of iOS 5.0 and above.

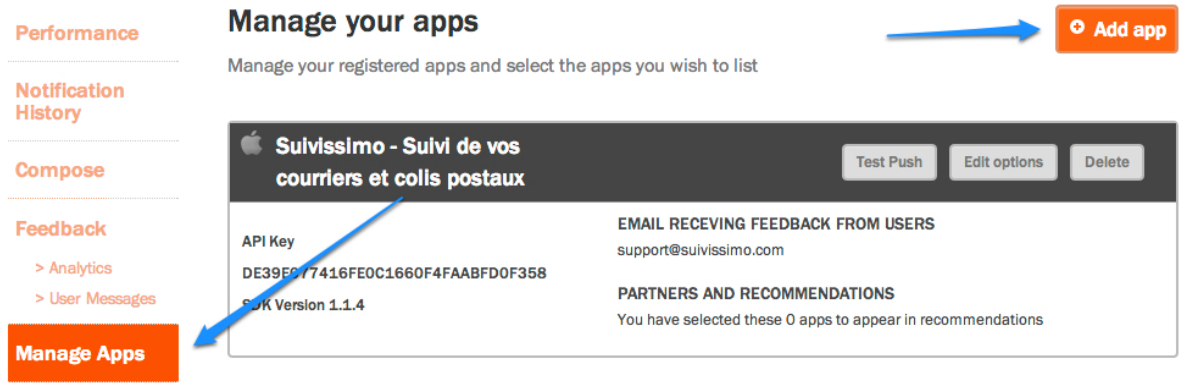
This SDK requires App Booster SDK 1.1.5 and above. You can find its own documentation in the appsfire dashboard : http://dashboard.appsfire.com/app/doc (you must be logged in).

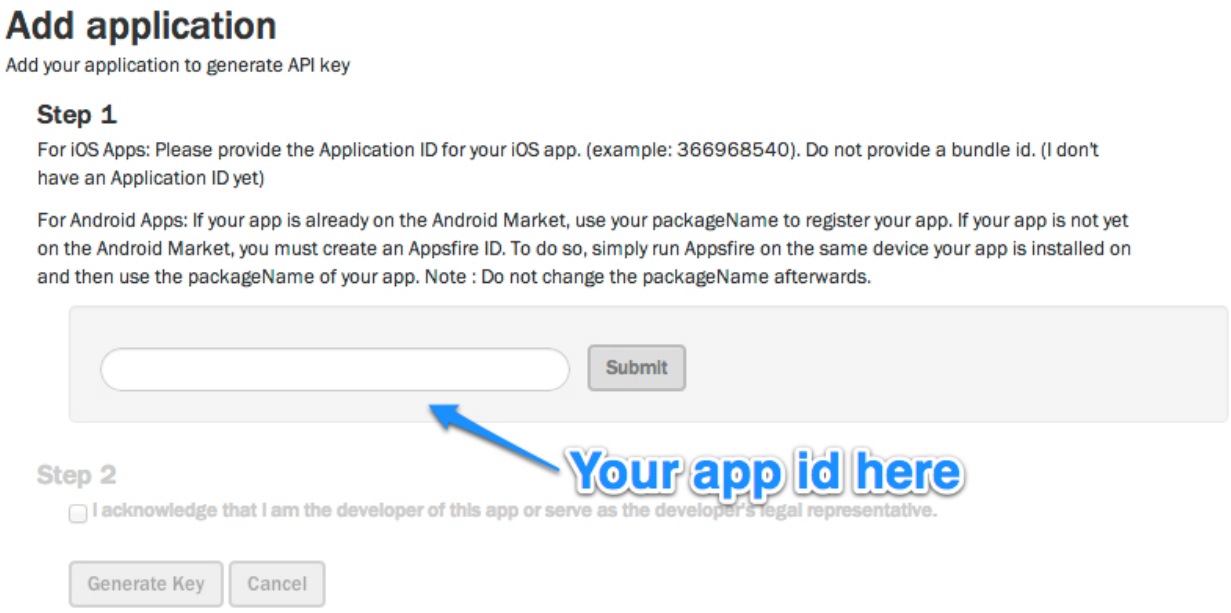
B. Creating your API key

This step is only necessary if you don't have App Booster SDK implemented yet. If you've already implemented App Booster, you may skip to Section C.

Please go to **http://dashboard.appsfire.com/** and log in.

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 in the next step.

C. Basic implementation

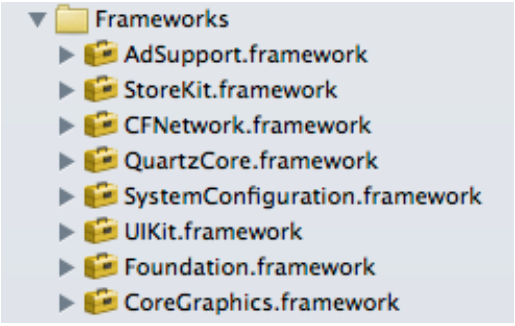
Step 1. Dropping the assets inside your project

The sample project displays the simplest way to install the SDK inside your app, by drag & dropping the folders « afadunit » and « appboostersdk » into your own project from the sample project.

Step 2. Making sure all frameworks are included

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

If you already have App Booster, you only need to add AdSupport and StoreKit.



Step 3. Add App Booster SDK

Note: You can skip this step if App Booster SDK is already implemented in your project.

To start, make sure you import the App Booster header file called “AFAppBoosterSDK.h”.

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

Then, inside the « didFinishLaunching » call-back method of your application delegate you need to call the « connectWithAPIKey » method.

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

    // connect app booster sdk
    [AFAppBoosterSDK connectWithAPIKey:@"INSERT YOUR API KEY HERE"];

    // [...]
    // window & root controller creation

    return YES;

}
```

For further details about the App Booster SDK installation, we recommend you to check its own documentation - http://dashboard.appsfire.com/app/doc

Step 4. Add Advertising SDK

To start, make sure you import the Ad Unit header file called “AFAdUnit.h”.

```
#import "AFAdUnit.h"
```

Then, inside the didFinishLaunching call-back method of your application delegate, you can define the various options:

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

    // connect app booster sdk
    [AFAppBoosterSDK connectWithAPIKey:@"INSERT YOUR API KEY HERE"];

    // here you can use the debug mode
    #if DEBUG
        [AFAdUnit setDebugModeEnabled:YES];
    #endif

    // decide to use in-app download when it's possible (optional)
    [AFAdUnit setUseInAppDownloadWhenPossible:YES];

    // tell ad unit to prepare (optional)
    // this method will be automatically called if you do a request
    [AFAdUnit prepare];


    // [...]
    // window & root controller creation

    return YES;

}
```

We recommend that you to call the « prepare » method as soon as possible so the library can prepare itself (and download the ads). In the worst case, this method will be called when you try to request an ad.

And finally, you can request a modal ad somewhere in your code. Please make sure to replace the only requested parameter with the proper UIViewController. Most of the time the following example should work. But it all depends on the architecture of your application.

```
[ AFAdUnit requestModalAdWithController:[ UIApplication
sharedApplication ].keyWindow.rootViewController ];
```

As there may not be an ad when you want it, there is a method to check if there is an ad available:

```
if ([AFAdUnit isThereAModalAdAvailable]) {
    NSLog(@"Ads are loaded and there is one available");
    // request a modal ad here
}
```

But that’s not necessary. Once you requested a modal ad, it’ll be added to a queue and presented as soon as ads are loaded. It depends how and when you want to present the ad.

We suggest you to read the next section which explains how to be alerted when ads are loaded (via the delegate).

D. Supplementary implementation

Step 1. Fully determine the general options

Using the in-app overlay

In iOS 6 it became possible to display an item's page in the App Store via a modal controller instead of redirecting the user to the App Store app. This is enabled by default in the Advertising SDK, as it serves you and your users by not kicking them out of your app.

If the user is on iOS 5, or if a problem occurs with the in-app overlay (this particular Apple API isn't very stable), then the user will be redirected to the App Store app.

You can modify the default value (`YES`) like this:

```
[AFAdUnit setUseInAppDownloadWhenPossible:YES];
```

Test your implementation via the « debug » mode

We added a debug mode because there may not be a campaign available when you implement the Advertising library in your application.

In debug mode:
- the SDK will display an app no matter what. Chances are it'll be the Appsfire app.
- there won't be any capping. Meaning that you'll be able to display the « app of the day » as much as you want, while in theory it should be once a day.

Be careful, though. Do not keep Debug Mode on in your app when it's submitted to Apple. You can enable this mode like this (we recommend that you to keep the #if #endif as a safeguard):

```
#if DEBUG
    [AFAdUnit setDebugModeEnabled:YES];
#endif
```

Step 2. Use the Ad Unit delegate

Prepare your class to support the delegate events

We recommend that you use the Ad Unit delegate for a better implementation of the library. First you need to add « AFAdUnitDelegate » to the @interface of the chosen class (most of the time it'll be the app delegate):

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

// properties & methods

@end
```

Don't forget to set this class as the delegate in the « didFinishLaunchingWithOptions » method:

```
// subscribe to events from ad unit (optional)
[AFAdUnit setDelegate:self];
```

So in the method it should look like this:

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

    // connect app booster sdk
    [AFAppBoosterSDK connectWithAPIKey:@"INSERT YOUR API KEY HERE"];

    // here you can use the debug mode
    #if DEBUG
        [AFAdUnit setDebugModeEnabled:YES];
    #endif

    // decide to use in-app download when it's possible (optional)
    [AFAdUnit setUseInAppDownloadWhenPossible:YES];

    // subscribe to events from ad unit (optional)
    [AFAdUnit setDelegate:self];

    // tell ad unit to prepare (optional)
    // this method will be automatically called if you do a request
    [AFAdUnit prepare];


    // [...]
    // window & root controller creation

    return YES;

}
```

Once this is done, you'll be able to receive various events:

All are optional, you can implement the ones you want. All these calls are done on the main thread.

1. When the library is initialized

This method isn't critical for a basic implementation. But you know that the library initialized correctly (but didn't receive any ad yet).

```
- (void)adUnitDidInitialize {

}
```

2. When the first modal ad is ready to be displayed after the library initialization

This method is very useful for a basic implementation. You can use it to fire an internal event in your application, so you know there is an ad to display.

```
- (void)modalAdIsReadyForRequest {

    // if you request the ad immediately, you can be sure that a modal ad is
available
    // you need to specify the UIViewController that will be used to display
the modal add & eventually the in-app overlay
    [AFAdUnit requestModalAdWithController:[UIApplication
sharedApplication].keyWindow.rootViewController];

    // if it's not the proper time, then you can request it a bit later

}
```

3. If a modal overlay fails to present

This method is not necessary in your application's life, but can be useful to debug an eventual problem.

It's called when the modal ad you requested could not be displayed. There are various reasons this could occur, and you can use the code in the « NSError » to understand why. The most common one is « AFAdUnitErrorCodeNoAd », meaning there was no ad to display (i.e., it's not a technical matter).

```
- (void)modalAdRequestDidFailWithError:(NSError *)error {

    NSLog(@"Ad Unit Request Failed = %@", error.localizedDescription);

    // optional, you can implement a reaction
    switch (error.code) {

        case AFAdUnitErrorCodeBadCall:
            break;
        case AFAdUnitErrorCodeNoAd:
            break;
        case AFAdUnitErrorCodeAdAlreadyDisplayed:
            break;
        default:
            break;

    }

}
```

4. When a modal ad is going to be presented

Depending on your implementation of AdUnit, you may want to cancel the presentation of the modal ad at the very last moment.

If you don't implement the following method, the modal ad will directly go on screen when ready. If you decide to cancel it because you are doing something important on screen, or if you aren't in a break-out session, then return `NO`

```
- (BOOL)shouldDisplayModalAd {

    return YES;

}
```

5. When the modal ad will appear / will disappear

It's always better to be alerted of tierce library actions. That's why we tell you when we are going to present the modal ad, and when it's going to be dismissed.

```
- (void)modalAdWillAppear {

    //

}

- (void)modalAdWillDisappear {

    //

}
```

Step 3. Full tour of Modal ad methods

Request a modal ad

As mentioned above, you can easily request a modal ad. You only need to pass a UIViewController parameter that will be used to present the modal, and eventually the in-app overlay.

Important: this method isn't necessarily synchronous. If ads aren't loaded when you call the method, your request will be added in a queue and processed a bit later.

```
[AFAdUnit requestModalAdWithController:[UIApplication
sharedApplication].keyWindow.rootViewController];
```

In our example we use the « rootViewController » of the « keyWindow » because it'll be correct for most of the applications. Be sure to indicate the proper UIViewController depending on your application.

Check if there is a modal ad to display

If you want to check the availability of the modal ad in a breakout session, there is a straightforward method for that. Not only it'll verify if the library is correctly initialized and ads are loaded from the web service, but it will also check if a modal ad can be displayed right now.

```
if ([AFAdUnit isThereAModalAdAvailable]) {

    // request a modal ad here

}
```

Cancel a pending request

Your request won't necessarily display a modal ad as soon as you request it (e.g., library isn't initialized, ads aren't loaded yet). In this case, your request will be added in a queue and will be processed at the right moment. If, in the meantime, you decide to abort any pending request, you can use the following method:

```
[ AFAdUnit cancelPendingAdModalRequest ];
```

If a modal ad is canceled, the method will return `YES` and you'll get a delegate event via the `modalAdRequestDidFailWithError:` method.

Check if a modal ad is currently displayed

We implemented it as an easy way for you to check if our library is currently displaying something on the screen.

```
BOOL isModalDisplayed = [ AFAdUnit isModalAdDisplayed ];
```

Note that you can know the start and the end of a modal being displayed via the Ad Unit delegate (cf step 2).

E. Find you problem step-by-step

Here is a checklist to help you find an eventual problem you could encounter while implementing the library into your application. Before contacting us, please check each point so we can help you faster in your debugging process.

1. Is the library correctly initialized?

Before anything, the library needs to initialize itself. Most of the time it'll be done in the first two seconds. But here's how to check if the process was successful:

```
if ([AFAdUnit isInitialized]) {

    NSLog(@"Ad Unit is initialized");

}
```

It's best to implement the delegate method which will let you know when the library initialized:

```
- (void)adUnitDidInitialize {

   NSLog(@"Ad Unit is initialized");

}
```

Why is it that the library doesn't initialize?
- The process isn't finished
- You haven't set an API key
- The device isn't connected to the internet

2. Are the ads loaded from the web service?

The second thing you should verify is the connection between the library and the web service. When the library is initialized, we try to fetch the latest ads from the web service. Once received, we'll process any eventual pending request.

Like for the library initialization, you can check if this process was successful. This doesn't mean there'll be an ad available, but only that the synchronization with the server was correctly done.

```
if ([AFAdUnit areAdsLoaded]) {

    NSLog(@"Ads correctly loaded from web service");

}
```

Why is it that the ads will not load?
- Library isn't initialized
- The process isn't finished
- The device isn't connected to the Internet

3. Is there a modal ad to display?

Now you are sure ads are correctly loaded, you can check if there is any modal ad available. As for the library initialization, you have two ways to know it.

The first one is pretty simple and can be done whenever you want. It's the same as explained in section D step 3:

```
if ([AFAdUnit isThereAModalAdAvailable]) {

    NSLog(@"A Modal Ad is Available");

}
```

It's best to implement the delegate method (section 2 step 3) which will let you know when ads are loaded and there is a modal ad available:

```
- (void)modalAdIsReadyForRequest {

    NSLog(@"A Modal Ad is Available");

}
```

Why is it that there is no modal ad available?
- Ads aren't loaded from the web service
- You aren't in « debug mode » and there is no ad to display today

If you want to test your application, be sure to be in « debug mode » that will display an app whatever the conditions. Just make sure to not enable it when submitting your app to Apple!

4. What if requesting a modal doesn't work?

First be sure to have implemented the delegate method that alerts you when a modal failed to display. It may give you a clue! Check the method « modalAdRequestDidFailWithError: » in section D, step 2.

If you verified that a modal ad is available, but nothing happens when you request it, then it's probably a problem with the UIViewController parameter that you are sending. By default our example uses the following controller: « `[UIApplication sharedApplication].keyWindow.rootViewController` » which is the main controller given to the window in the app initialization. But depending on the age of your application (i.e., the iOS version in which it was created), or even its architecture, this may not be the best choice.

In summary, verify that the controller is appropriate. How? Try to present an empty controller to see if something happens on the screen. If not, then it's definitely not the right one.

5. Still a problem?

Don't hesitate to contact us (see « Support Considerations » section), we'll do our best to help you.

F. Frequently Asked Questions

How many times a day can the « modal ad » be displayed?

In a normal state, the « modal ad » will be displayed once a day. For example, if an user sees an ad at 11pm, he'll be able to see the next one at midnight.
However, to help you implementing the library in your application, the « debug » mode won't be affected by this capping system.

What happens if a modal ad is being displayed and app goes to background?

We dismiss the ad and the eventual in-app overlay that were displayed.

G. Support Considerations

Please contact jonathan@appsfire.com for general inquiries. For a specific technical question, add vincent@appsfire.com in the email

High-level inquiries should be directed to both yann@appsfire.com and ouriel@appsfire.com