Section 3 - Advanced push configuration

iOS iOS 5.2.x and earlier iOS 5.3.x iOS 5.4.x iOS 5.5.x iOS 5.6.x iOS 6.0.x iOS 6.1.x iOS - 6.3.x Latest version - 6.4.x

> **Have a splashcreen in your app ?**
> Allowing a content to be displayed in your splashcreen can affect the push behaviour.
>
> For this reason, a method exists to prevent the display of any Accengage content once the splashcreen is loading. Please check this page for more information.

3.1 Customize the SDK default launching configuration

iOS iOS 5.2.x and earlier iOS 5.3.x iOS 5.4.x iOS 5.5.x iOS 5.6.x iOS 6.0.x iOS 6.1.x iOS - 6.3.x Latest version - 6.4.x
The library provides a number of advanced configuration features that can be controlled with the `ACCConfiguration` object.

The configurations can be set simply by using the **AccengageConfig.plist** you've added to your project, or at runtime using the exposed properties of the `ACCConfiguration` object. See the `ACCConfiguration` documentation for the full list of configuration options.

IDFA collection

The **IDFA (Identifier for Advertisers)**, is automatically collected by the library. Note that you can disable the **IDFA** collection by setting the property to `NO`.

Automatic integration

It's not required to implement the notification-related delegate methods. The library can intercept the UIApplicationDelegate and the UNUserNotificationCenterDelegate messages and forward them to your original delegate.

This should be compatible with the majority of application use cases, but if you prefer, you can disable this behavior.

If you choose to disable the automatic integration, you will need to implement all notification-related `UIApplicationDelegate` and `UNUserNotificationCenterDelegate` methods. See the Push documentation for more details.

Tracking mode

Depending on your privacy rules, you can decide to choose the restricted tracking mode which disables the collection of certain device data such as connection type, carrier name, local date and jailbreak state.

Note that if you use the restricted mode, some advanced targeting features may not be available. This restriction is not recommended for most integrations.

Session timeout

The library provides automated session management. This value indicates how long (***in seconds***) the app may be in the background before starting a new session upon resume. A default implementation has a session timeout period of **5 minutes**. If the app remains in the background for longer than the session timeout period, the next time it's opened a new session will be started.

> Note
>
> Each new session equals to a new application visit which means that new server calls are made.
>
> It is for this reason that we recommend that you keep the session timeout period set to 5 minutes.

Delay SDK Launch

If you prefer to start the SDK with a delay (e.g. as a result of a user action), you'll need to pass the launch options information to the SDK.

This information is provided by the system in the didFinishLaunchingWithOptions callback and should be set using the launchOptions property of the ACCConfiguration object.

Custom notification sounds

You can specify a custom sound, that iOS will play when it receives a local or remote notification for your app. For more details, you can consult Apple's documentation.

Remember that the default sound is played when:

- The sound file is not found.
- The sound is longer than 30 seconds.
- The data format is invalid. Accepted data formats are: aiff, wav, or caf.

To add the sound file to your project, you can simply drag and drop it to your resource folder. Make sure you check the box Copy items if needed.

In the Accengage dashboard, you can easily specify this sound when composing a push message by selecting "custom" in the sound section, and enter the file name with its extension (customsound.wav for instance).

> Prefer a short file name.

Configure interactive notifications

Interactive notifications give the user a quick and easy way to perform relevant tasks in response to a notification. Instead of the user being forced to launch your app, the interface for an actionable notification displays custom action buttons that the user can tap. When tapped, each button dismisses the notification interface and forwards the selected action to your app for immediate handling. Forwarding the action to your app avoids the need for the user to navigate further in your app to perform the action, thereby saving time.

A set of default notification categories is provided by Accengage. If you want to add more categories for interactive notifications, you should call the `setCustomCategories:` before registering for push notifications:

Objective-C

```
// The app custom categories set
[[Accengage push] setCustomCategories:customCategories];

// Register for notification
ACCNotificationOptions options =
(ACCNotificationOptionSound|ACCNotificationOptionBadge|ACCNotificationOptionAl
ert|ACCNotificationOptionCarPlay);

[[Accengage push] registerForUserNotificationsWithOptions:options];
```

Swift

```
// The app custom categories set
Accengage.push().customCategories = customCategories

// Register for notification
Accengage.push().registerForUserNotifications(options: [.sound, .badge,
.alert, .carPlay])
```

> Note

If you would like to handle custom categories and buttons, you must declare them in the Accengage Dashboard (Settings > Settings > Categories (Buttons)). See our User Guide for more details.

3.2 Handling push delegate callbacks manually

iOS iOS 5.2.x and earlier iOS 5.3.x iOS 5.4.x iOS 5.5.x iOS 5.6.x iOS 6.0.x iOS 6.1.x iOS - 6.3.x Latest version - 6.4.x
If you choose to disable the automatic integration, by changing the **automaticPushDelegateEnabled** property of your **ACCConfiguration** ob ject to **NO**, you will need to add the following code:

1. Set `UNUserNotificationCenter` delegate

Objective-C

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

    // Set up the UNUserNotificationCenter delegate
    [UNUserNotificationCenter currentNotificationCenter].delegate = self;
}
```

Swift

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {

    // Set up the UNUserNotificationCenter delegate
    UNUserNotificationCenter.current().delegate = self
}
```

2. Implement the delegates methods

Objective-C

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(nonnull NSData
*)deviceToken {

    [[Accengage push]
didRegisterForUserNotificationsWithDeviceToken:deviceToken];
}

- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo
fetchCompletionHandler:(nonnull void
(^)(UIBackgroundFetchResult))completionHandler {

    [[Accengage push] didReceiveRemoteNotification:userInfo];
    completionHandler(UIBackgroundFetchResultNoData);
}
```

```
- (void)application:(UIApplication *)application
handleActionWithIdentifier:(NSString *)identifier
forRemoteNotification:(nonnull NSDictionary *)userInfo
completionHandler:(nonnull void (^)())completionHandler {

    [[Accengage push] handleActionWithIdentifier:identifier
forRemoteNotification:userInfo];
    completionHandler();
}

- (void)application:(UIApplication *)application
handleActionWithIdentifier:(NSString *)identifier
forRemoteNotification:(nonnull NSDictionary *)userInfo
withResponseInfo:(nonnull NSDictionary *)responseInfo
completionHandler:(nonnull void (^)())completionHandler {

    [[Accengage push] handleActionWithIdentifier:identifier
forRemoteNotification:userInfo withResponseInfo:responseInfo];
    completionHandler();
}

- (void)application:(UIApplication *)application
didReceiveLocalNotification:(nonnull UILocalNotification *)notification {

    [[Accengage push] didReceiveLocalNotification:notification];
}

- (void)application:(UIApplication *)application
handleActionWithIdentifier:(NSString *)identifier
forLocalNotification:(nonnull UILocalNotification *)notification
completionHandler:(nonnull void (^)())completionHandler {

    [[Accengage push] handleActionWithIdentifier:identifier
forLocalNotification:notification];
    completionHandler();
}

- (void)application:(UIApplication *)application
handleActionWithIdentifier:(NSString *)identifier
forLocalNotification:(nonnull UILocalNotification *)notification
withResponseInfo:(nonnull NSDictionary *)responseInfo
completionHandler:(nonnull void (^)())completionHandler {

    [[Accengage push] handleActionWithIdentifier:identifier
forLocalNotification:notification withResponseInfo:responseInfo];
    completionHandler();
}

// UNUserNotificationCenterDelegate methods

- (void) userNotificationCenter:(UNUserNotificationCenter *)center
willPresentNotification:(UNNotification *)notification
withCompletionHandler:(void
(^)(UNNotificationPresentationOptions))completionHandler {

    completionHandler([[Accengage push]
willPresentNotification:notification]);
}
```

```
- (void) userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response
withCompletionHandler:(void (^)())completionHandler {
```

```
    [[Accengage push] didReceiveNotificationResponse:response];
    completionHandler();
}
```

Swift

```
func application(_ application: UIApplication,
didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
        Accengage.push()?.didRegisterForUserNotifications(withDeviceToken:
deviceToken)
}

func application(_ application: UIApplication, didReceiveRemoteNotification
userInfo: [AnyHashable : Any], fetchCompletionHandler completionHandler:
@escaping (UIBackgroundFetchResult) -> Void) {
 Accengage.push()?.didReceiveRemoteNotification(userInfo)
 completionHandler(.noData)
}
func application(_ application: UIApplication, handleActionWithIdentifier
identifier: String?, forRemoteNotification userInfo: [AnyHashable : Any],
completionHandler: @escaping () -> Void) {
 Accengage.push()?.handleAction(withIdentifier: identifier!,
forRemoteNotification: userInfo)
 completionHandler()
}

func application(_ application: UIApplication, handleActionWithIdentifier
identifier: String?, forRemoteNotification userInfo: [AnyHashable : Any],
withResponseInfo responseInfo: [AnyHashable : Any], completionHandler:
@escaping () -> Void) {
 Accengage.push()?.handleAction(withIdentifier: identifier,
forRemoteNotification: userInfo, withResponseInfo: responseInfo)
 completionHandler()
}

func application(_ application: UIApplication, didReceive notification:
UILocalNotification) {
 Accengage.push()?.didReceive(notification)
}

func application(_ application: UIApplication, handleActionWithIdentifier
identifier: String?, for notification: UILocalNotification,
completionHandler: @escaping () -> Void) {
 Accengage.push()?.handleAction(withIdentifier: identifier, for:
notification)
 completionHandler()
}

func application(_ application: UIApplication, handleActionWithIdentifier
identifier: String?, for notification: UILocalNotification,
withResponseInfo responseInfo: [AnyHashable : Any], completionHandler:
@escaping () -> Void) {
 Accengage.push()?.handleAction(withIdentifier: identifier, for:
notification, withResponseInfo: responseInfo)
 completionHandler()
```

```
}

// UNUserNotificationCenterDelegate methods

func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent
notification: UNNotification, withCompletionHandler completionHandler:
@escaping (UNNotificationPresentationOptions) -> Void) {
 completionHandler((Accengage.push()?.willPresent(notification))!)
}

func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive
response: UNNotificationResponse, withCompletionHandler completionHandler:
@escaping () -> Void) {
```

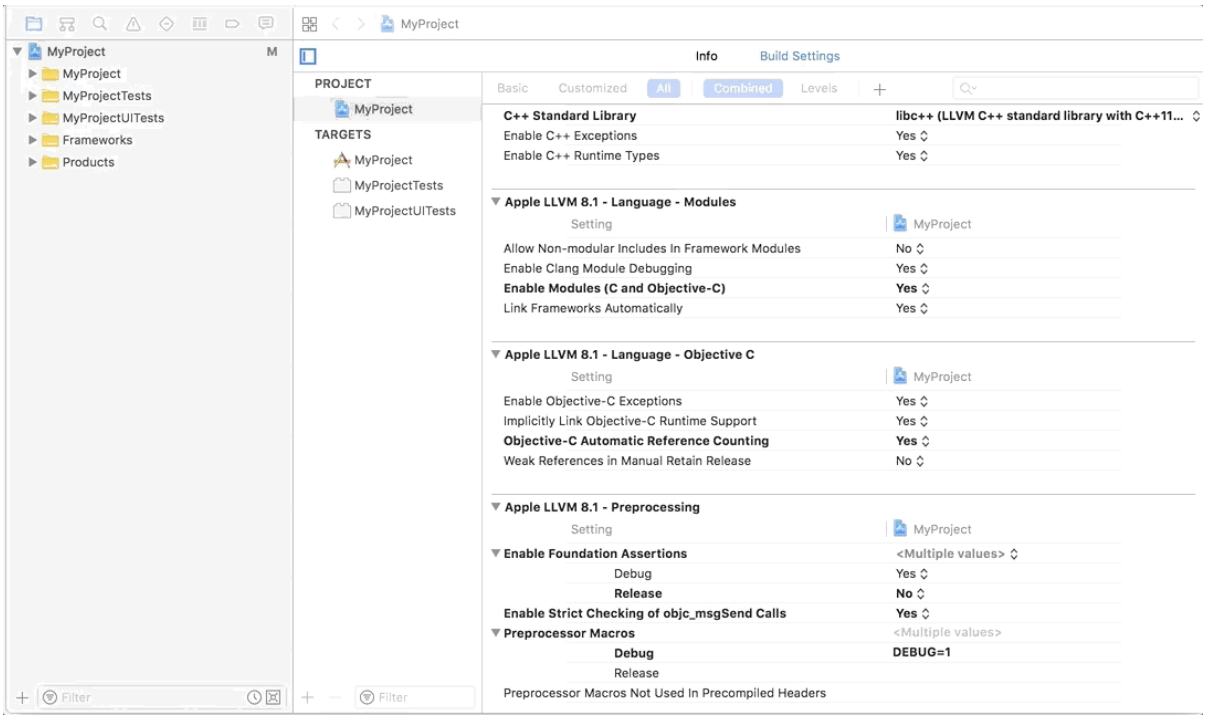
```
 Accengage.push()?.didReceive(response)
 completionHandler()
}
```

3.3 Carousel Template

iOS iOS 5.2.x and earlier iOS 5.3.x iOS 5.4.x iOS 5.5.x iOS 5.6.x iOS 6.0.x iOS 6.1.x iOS - 6.3.x Latest version - 6.4.x

> Carousel template is a rich notification template so before starting please make sure you are able to receive push with media attachments. If not, please refer to the Media Attachments section.

To enable the Carousel template in your project, you will need to create a Notification Content Extension.



> **Deployment target**
> By default, XCode set the most recent iOS version available as deployment target. If you run your application on a device that has an older iOS version, push notifications with carousel template won't be able to work. We recommend to set 10.0 as deployment target.

Cocoapods

1. In your Podfile, add the pod *Accengage-iOS-SDK-Extension* to your newly created target :

```
# Add it to your Podfile
target 'YOUR_NOTIFICATION_CONTENT_EXTENSION_TARGET' do
    pod 'Accengage-iOS-SDK-Extension', '~> 1.1.0'
end
```

2. Then, update your Cocoapods project.

```
$ pod update
```

Manual integration

Link the `AccengageExtension.framework` to your newly created target

Then, change the newly created NotificationViewController class in such a way that it inherits from `ACCNotificationContentViewController` and replace the code created by XCode by the following one :

Objective C

```
 // NotificationViewController.h
#import <AccengageExtension/AccengageExtension.h>

@interface NotificationViewController : ACCNotificationContentViewController
@end
```

```
 // NotificationViewController.m
#import "NotificationViewController.h"
#import <UserNotifications/UserNotifications.h>
#import <UserNotificationsUI/UserNotificationsUI.h>

@implementation NotificationViewController
@end
```

Swift

```
import AccengageExtension
import UserNotifications
import UserNotificationsUI

class NotificationViewController: ACCNotificationContentViewController {
}
```

In the NotificationViewController scene of the file newly created *MainInterface.storyboard* file, delete the default *Hello world* label.

Finally, in the **Info.plist** of the extension, update `UNNotificationExtensionCategory` to `acc_carousel_category`

3.4 Prevent notification tracking

iOS iOS 5.2.x and earlier iOS 5.3.x iOS 5.4.x iOS 5.5.x iOS 5.6.x iOS 6.0.x iOS 6.1.x iOS - 6.3.x Latest version - 6.4.x

Sometimes, it might be useful to prevent push notifications from performing some tracking, for various reasons : avoid displaying web redirection on top of ads or during loading splashscreen, respecting user preferences etc...

Accengage provides you a way to disable such tracking.

You can disable or enable the tracking of push notification by suspending the push service :

Objective-C

```
[Accengage push].suspended = YES; // or NO to reenable tracking
```

Swift

```
Accengage.push().suspended = true // or false to reenable tracking
```

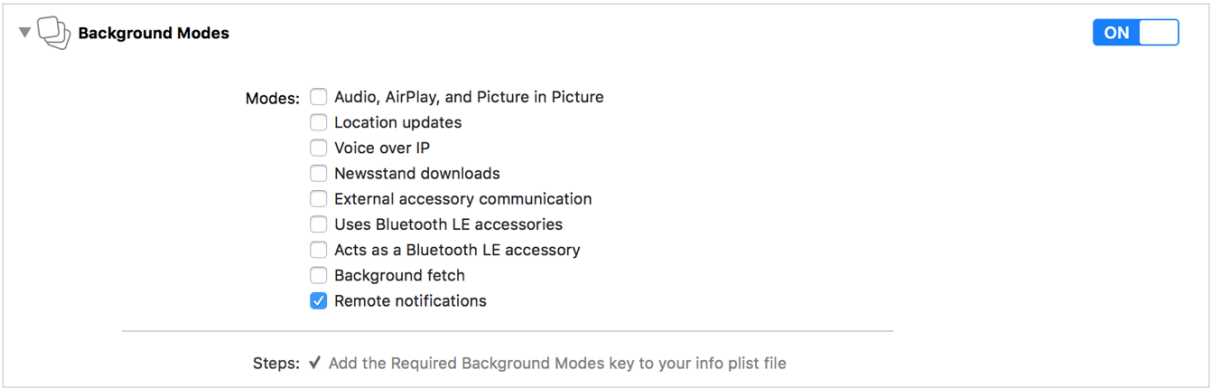
> Note
>
> This won't affect In-App messages tracking.

Silent push notifications

Silent push notifications improve the user experience by giving you a way to wake up your app periodically so that it can refresh its data in the background.

To enable silent push in your Xcode project:

1. In the Project Navigator, select your project.
2. In the editor, select your iOS app target.
3. Select the Capabilities tab.
4. Enable the Background Modes capability.
5. Enable the Remote notifications background mode.



To enable silent push on the Accengage dashboard, tick the content available checkbox when configuring your push message.

Please bare in mind that APNs treats silent push notifications as low priority and may throttle their delivery altogether if the total number becomes excessive. The actual limits are dynamic and can change based on conditions, but try not to send more than a few notifications per hour.