Integration

Add the Xamarin SDK to your reference

On both project Android/iOS of your own project you will need to add as reference the Accengage Plugin available on NuGet.

Right click on Reference (on both iOS/Android side) and choose Manage NuGet Package, then search Xamarin.Accengage SDK and add it.

Then we recommend you to create a class in a SharedProject which will allow you to only have one class to handle the majority of methods

you need to run properly our SDK, on iOS and Android side at the same time, since the SharedProject will search the method and adapt on the

environnement he is in to choose the working one.

Using

Using AccengageSDK at the top of your AppDelegate (iOS) or Activity (And) and any of your class:

```
using AccengageSDK;
```

Android Integration

Prerequisites

- Compile using AndroidVersion 6

To use all the potential of the android SDK, some additionnal integration step needs to be done.

Put our service tag insinde your application tag with the proper senderId, partnerID and privateKey.

AndroidManifest.xml

```
<service android:name="com.ad4screen.sdk.A4SService" android:label="A4S Service"
android:process=":A4SService">
 <meta-data android:name="com.ad4screen.partnerid" android:value="partnerId" />
 <meta-data android:name="com.ad4screen.privatekey" android:value="privateKey" />
 <meta-data android:name="com.ad4screen.senderid" android:value="gcm:senderId" />
</service>
```

Inside all your Activities, you will need to Sub Classing any Activity Type.

Following permissions are required in your manifest :

AndroidManifest.xml

```
<permission android:name="your-package.permission.C2D_MESSAGE"
android:protectionLevel="signature" />
<permission android:name="your-package.provider.A4S_READ_DATABASE"
android:protectionLevel="signature"/>
<permission android:name="your-package.provider.A4S_WRITE_DATABASE"
android:protectionLevel="signature"/>
<uses-permission android:name="your-package.permission.C2D_MESSAGE"/>
```

Finally you need to add in your manifest inside the application tag our GCMHandler,

AndroidManifest.xml

```
<receiver android:name="com.ad4screen.sdk.GCMHandler"
android:permission="com.google.android.c2dm.permission.SEND">
 <intent-filter>
     <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="${applicationId}" />
    </intent-filter>
</receiver>
```

A4SProvider,

AndroidManifest.xml

```
<provider android:authorities="your-package.ad4screen.provider"
 android:readPermission="your-package.provider.A4S_READ_DATABASE"
    android:writePermission="your-package.provider.A4S_WRITE_DATABASE"
    android:name="com.ad4screen.sdk.provider.A4SProvider"
android:process=":A4SService" />
```

and NotificationActionsReceiver:

AndroidManifest.xml

```
<receiver android:name="com.ad4screen.sdk.NotificationActionsReceiver"
    android:process=":A4SService">
 <intent-filter>
     <action android:name="com.ad4screen.sdk.notification.CLICK" />
  <action android:name="com.ad4screen.sdk.notification.action.CLICK" />
  <category android:name="${applicationId}" />
 </intent-filter>
</receiver>
```

Plugin

Prerequisites

To use all of our plugins you will need to setup your compiled android version to 7 since google play services 9.6.1 used the Android.Support.v4 24+

Xamarin.Accengage.Android.Plugin Firebase v1.0.0 :

This plugin allows the SDK to register/unregister to Firebase Cloud Messaging (FCM).

As of April 10, 2018, Google has deprecated Google Cloud Messaging (GCM). This gateway will no longer be available to broadcast Push Notifications to your users as of April 11, 2019.
We highly recommend you to migrate your GCM project to Firebase Cloud Messaging (FCM).

You cannot use our plugin Firebase and our plugin Play Services Push at the same time.

Xamarin.Accengage.Android.Plugin Play Services v1.2.0 :

This plugin allows the SDK to:

- Register/Unregister to GCM using GooglePlayServices (official and recommended Google's method)
- Retrieve and send to Accengage the Google Advertiser ID (IDFA) automatically
- Geolocation using GooglePlayServices (SDK 3.1.0+). Usefull about power comsumption.
- Geofencing using GooglePlayServices (SDK 3.1.0+)

This plugin regroup five plugins which you can import in your project separately in fonction of what you need :

- Xamarin.Accengage.Android.Plugin Play Services Base v1.2.0
- Xamarin.Accengage.Android.Plugin Play Services Location v1.2.0
- Xamarin.Accengage.Android.Plugin Play Services Push v1.2.0
- Xamarin.Accengage.Android.Plugin Play Services Geofence v1.2.0
- Xamarin.Accengage.Android.Plugin Play Services AdvertiserId v1.2.0

Xamarin.Accengage.Android.Plugin Beacon v1.2.0

This plugin allows the SDK to:

- Listen for Beacons
- Trigger In-Apps and Local Notifications using Beacons

Xamarin.Accengage.Android.Plugin Badger v1.2.0

This plugin allows the SDK to:

- Manage badge on all layer

iOS Integration

To use all the potential of the iOS `SDK`, some additionnal integration step needs to be done.

Prerequisites

- XCode 8
- iOS 8 and higher.

Configure push notifications

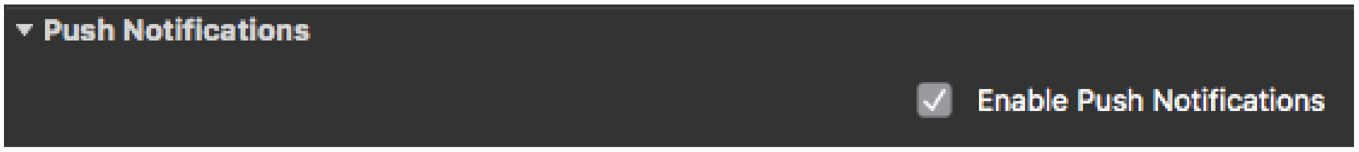
Push notifications allow an app that isn’t running in the foreground to notify the user when it has information for them. Push notifications originate from a notification server that you manage and are pushed to your app on a user’s device by the Apple Push Notification service (APNs).

First, you enable push notifications in the Xamarin project.

After you enable push notifications, generate and export a TLS certificate.

Enable push notifications

Please ensure that in the Entitlements.plist file, your Push Notifications entitlement is checked.



Enable Push Notifications [✓]

Creating and using certificates

Please follow this Xamarin documentation link to learn how to create a certificate, associate it with a provisioning profile, and then get a Personal Information Exchange certificate.

Configure URL schemes

Creating an URL schemes

Deep Linking is becoming very important, it's a great way to enhance the effectiveness of you app's marketing campaigns.

It will allow you to open a specific view of your app from another app, from a push message, from an in-app message

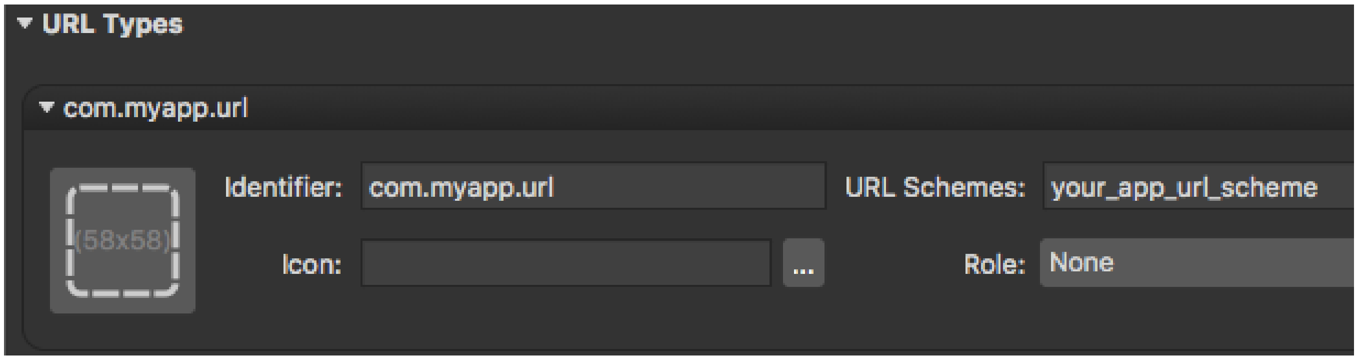
or even from a website to your app.

To take advantage of it, you must create a custom URL Scheme associated with your application. A URL scheme lets you

communicate with other apps through a protocol that you define. To find out more, check out Apple documentation.

To register your URL Scheme:

1. Open your Info.plist and select the Advanced tab.
2. Click Add URL Type in the URL Types expander.
3. In URL Schemes field, enter your_app_url_scheme.
4. Assign a unique identifier to the scheme to ensure uniqueness and avoid name collisions, so we'll use com.myapp.url for the example.



Identifier: com.myapp.url
URL Schemes: your_app_url_scheme
Icon:
Role: None

To verify that everything works:

1. Run your app.
2. Open Safari on the same device.

Type your_app_url_scheme:// in the address bar, then press Go.

Whitelisting URL Schemes

Like ATS, iOS 9 introduces changes that impact URL Scheme management. For more details, you can consult Apple's documentation.

Make sure that your application Info.plist includes the LSApplicationQueriesSchemes set with the Log App's URL scheme.

```
<key>LSApplicationQueriesSchemes</key>
<array>
 <string>your_app_url_scheme</string>
    <string>other_app</string>
</array>
```

Configure logging

Configure logging application

Accengage provide an application to help your team to identify for sure their device profile among the others, which is helpful for push notifications tests. It also allows our technical team to properly investigate in case of unexpected behavior. We strongly recommend it.

Note
This configuration is mandatory if Accengage teams need to test the SDK integration.

In order to be able to use our application, you will need to:

1. add the scheme bma4sreceiver to the schemes whitelist as explained in the previous section.
2. add a localhost exception to the App Transport Security key in your **info.plist**.

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <key>localhost</key>
        <dict>
            <key>NSExceptionAllowsInsecureHTTPLoads</key>
            <true/>
        </dict>
    </dict>
</dict>
```

Enabling console logs

For diagnostic purposes, the library contains an option that turns on logging. These logs allow you to gain more insight on the library's integration and ensure it's working as expected. These logs can also be helpful for support staff who are troubleshooting problems.

```
Accengage.SetLoggingEnabled(true)
```

Configuration File

First, get an empty configuration file to add to your project from this link.

Drag the AccengageConfig.plist file you just downloaded into the root of your Xamarin iOS project and add it to all targets then complete it with your own partner id and private key.

Right-click the file and set Build Action to BundleRessource.

Initialization

Now you're ready to begin implementing. Start the library in your AppDelegate FinishedLaunching method. From version 1.2.0 of the plugin, several ways to manager SDK starting are available depending on whether you need to use features related to GDPR. If you need more details, please refer to iOS native documentation.

```
public override bool FinishedLaunching (UIApplication application, NSDictionary
launchOptions)
{
 // Start the SDK with the configuration set in the AccengageConfig.plist file
 Accengage.Start();

 // Start the SDK with or without GDPR support and the configuration set in the
AccengageConfig.plist file
 Accengage.StartWithOptin(AccengageIOS.ACCOptIn.Enabled); // or
AccengageIOS.ACCOptIn.Disabled if you don't want to manage GDPR.

  // You also have the possibility to start with a start configuration setted
programmatically.
  AccengageIOS.ACCConfiguration configuration = new AccengageIOS.ACCConfiguration();
    configuration.AppId = "YOUR_APP_ID";
    configuration.AppPrivateKey = "YOUR_PRIVATE_KEY";

    Accengage.StartWithConfig(configuration);
    Accengage.StartWithConfigOptin(AccengageIOS.ACCOptIn.Enabled, configuration); //
or AccengageIOS.ACCOptIn.Disabled
}
```

GDPR management
If you plan to start the SDK using GDPR related features, data collection is disabled by default. Data collection is mandatory to use most of the SDK features. If you need it, you have to enable it explicitly.

```
// Enable or disable data collection.
 Accengage.SetOptinData(true) // or false
```

Plugin 1.2.0 also brings features related to geolocation optin.

```
// Enable or disable geolocation optin. Note that optin data enabled is mandatory to
allow geolocation optin.
 Accengage.SetOptinGeoloc(true) // or false
```

Import natives methods

If a feature is not available in the wrapper, You can directly use the binded sdk as if you coded natively. Use this namespace.

Android

```
using Com.Ad4screen.Sdk;
```

iOS

```
using AccengageIOS;
```