Xamarin

Getting Started

Accengage SDK allows you to track the execution and display of In-App and to handle Push notifications of your application.

This Xamarin documentation explains the various methods available and how to use them.

In order to understand more deeply the features of the SDK, please refer also to the native iOS and Android documentations.

Plugin Version 1.3.0:

- Android SDK version 3.8.4
- iOS SDK version 6.4.0

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.

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

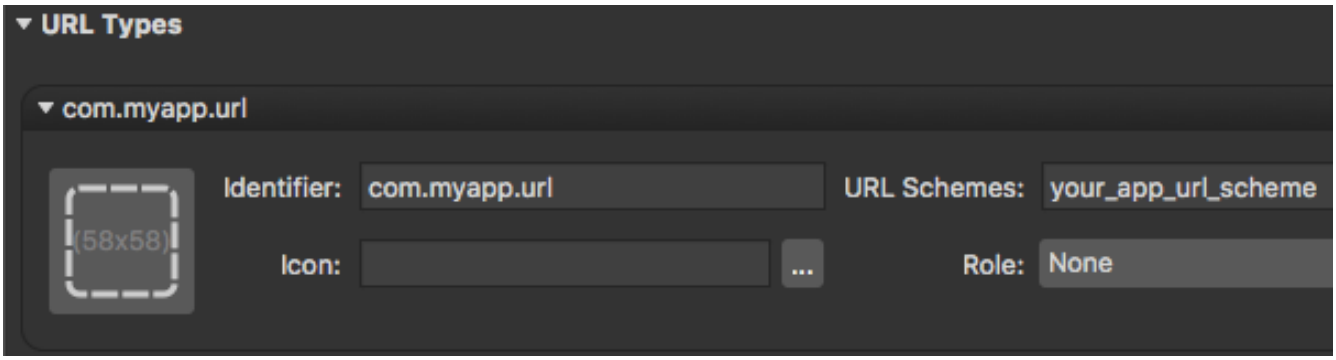
or even from a website to your app.

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.



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;
```

Push

Android

To enable push follow our android documentation to create a FCM project if you don't have one. Otherwise, follow our android documentation to migrate your GCM project to FCM.

Prevent Rich Push Notifications display

You can prevent the display of any RichPush notification, by calling:

```
A4S.Get(Context).PushNotificationLocked = true;
```

To reenable the RichPush Notifications, you can call:

```
 A4S.Get(Context).PushNotificationLocked = false;
```

You can enable/disable RichPush notifications at any time.

iOS

Registering

In order to register for user notifications, you can call the registerForUserNotificationsWithOptions() method. This means that you're no longer required to maintain the registration by yourself, just call this method and the library will request notification authorization for you.

Important
Your must register for user notifications every time your app is launched.

```
// Register for notification
AccengageIOS.ACCNotificationOptions options =
(AccengageIOS.ACCNotificationOptions.Alert |
AccengageIOS.ACCNotificationOptions.Badge |
AccengageIOS.ACCNotificationOptions.Sound);

AccengageIOS.Accengage.Push.RegisterForUserNotificationsWithOptions(options);
```

The Xamarin plugin supports iOS 12 provisional authorization. You can register this way by using following options :

AccengageIOS.ACCNotificationOptions.AuthorizationOptionProvidesAppNotificationSettings

AccengageIOS.ACCNotificationOptions.NotificationOptionProvisional

```
// You can add these parameters to register options.
options |=
AccengageIOS.ACCNotificationOptions.AuthorizationOptionProvidesAppNotificationSet
tings; // to activates the provisional authorization.
options |= AccengageIOS.ACCNotificationOptions.NotificationOptionProvisional; //
to indicates that the system should display a button for notification settings.

AccengageIOS.Accengage.Push.RegisterForUserNotificationsWithOptions(options);
```

You can get more informations about this feature in the native iOS SDK documentation.

Media attachments

iOS 10 introduces support for rich notifications, it adds the abitity to send push with media attachments such as images, sounds and videos.

> Rich notifications are available on iPhone 5s or later, iPad Pro, iPad Air or later, and iPad mini 2 or later.

To enable this functionality, you will need to create a Notification Service Extension.

> **Important**
> Only work with Xamarin.iOS 10.4 and higher.

1. Once your notification service is created, add the nuget package Xamarin.Accengage.iOS.Extension in extension.
2. Inherit from ACCNotificationServiceExtension in NotificationService and delete all the code generated by Xamarin.

```
using Foundation;
using AccengageExtensionIOS;

namespace AccengageNotificationService
{
    [Register("NotificationService")]
    public class NotificationService : ACCNotificationServiceExtension {}
}
```

It is possible to disable the display of push using :

```
SetPushServiceDisabled(true)
```

InApp

The Accengage SDK provides you with notifications display without any additional code to write in order to support them.

It is possible to disable the display of InApp, for example if you are using splashscreen or if you have views in which you don't want them to appear using.

```
SetInAppDisplayDisabled(true)
```

Tracking

Events

> To use the additional parameters of the custom events for targeting purposes, they must formatted as a valid json string. Use a timestamp to represent a date.

If you want to track a specific event :

```
TrackEvent(1000,"{\"string-type-field\":\"value\",\"bool-type-field\":true,\"number-type-field\":7201,\"date-type-field\":1498233848}")
```

Where 1000 is the eventId.

Custom Events

It is possible to build custom events and track them via the following method : Accengage.TrackCustomEvent(long type, dictionary<string, object> parameters);

CustomEvent type is supposed to be a number strictly higher than 1000 (long-typed).

CustomEvent parameters are designed to be a dictionary composed of key/value couples.

Keys are strings regardless of whether you are on iOS or Android

On the other hand, values type change depending on whether you are on iOS or Android platform.

Android

In Android we are expecting to pass value in the following types :

- Boolean
- Int32
- Int64
- float
- Double
- Java.Util.Date
- String

If you try to pass other kind of data, it will be treated as a String

```
Dictionary<string, object> parameters = new Dictionary<string, object>();
parameters["keyBoolean"] = true;
parameters["keyInteger"] = 23;
parameters["keyDouble"] = 2.0;
parameters["keyDate"] = new Java.Util.Date();
parameters["keyString"] = "Test String";
Accengage.TrackCustomEvent(1001, parameters);
```

iOS

In iOS we are expecting to pass value in the following types from the Foundation Framework :

- NSDate
- NSNumber

- NSString
- bool

If you try to pass other kind of data, it will be ignored.

```
Dictionary<string, object> parameters = new Dictionary<string, object>();
parameters.add("keyNSString", new NSString("foo bar"));
parameters.add("keyNSNumber", new NSNumber(2));
parameters.add("keyBool", true);
Accengage.TrackCustomEvent(1001, parameters);
```

Analytics

You can track specific events like “Add to Cart”, “Purchase” and “Lead”. Here is how to use each of these events.

Lead

```
TrackLead("lead label", "lead value");
```

Add to Cart

First of all you will need to create an Item using the AccengageItem class.

```
AccengageItem accengageItem = new
AccengageItem("idItem","labelItem","categoryItem",19.50,"EUR",1)
```

> *Currency should be a valid 3 letters ISO4217 currency (EUR,USD,..) and the 2 last arguments are price and quantity*

Now you just have to add it to the Cart.

```
TrackCart("cartId",  accengageItem, "EUR");
```

Purchase

```
TrackPurchase("purchaseId", "EUR", 19.5, listItem);
```

Update Device Info

You can create a device profile for each device in order to qualify the profile (for example, registering whether the user is opt in for or out of some categories of notifications).

A device profile is a set of key/value that are uploaded to Accengage server. In order to update information about a device profile, add the following lines to your code:

```
UpdateDeviceInfo(listUdi);
```

Where listUdi is a List<KeyValuePair<string, string>> containign all the informations to update.

New API to Update Device Information

You have the ability to use update device information using the following method : public static void UpdateDeviceInformation(string action, string key, object value);

Action are string-typed. Possible values are “set”, “delete”, “increment” and “decrement”. Using other values would be ineffective.

Android

In Android we are expecting to pass value in the following types :

- Boolean
- Int32
- Int64
- Double
- Java.Util.Date
- String

If you try to pass other kind of data, it will be treated as a String

```
Accengage.UpdateDeviceInformation("set", "firstname", "John");
Accengage.UpdateDeviceInformation("set", "lastname", "Doe");
Accengage.UpdateDeviceInformation("set", "age", 35);
Accengage.UpdateDeviceInformation("increment", "number_of_visit", 1.0);
Accengage.UpdateDeviceInformation("decrement", "article_left", 1.0);
```

> You can only operate “increment” or “decrement” actions on suitable value Double in Android

iOS

In iOS we are expecting to pass value in the following types from the Foundation Framework :

- NSDate
- NSNumber
- NSString
- bool

If you try to pass other kind of data, it would be ignored.

```
Accengage.UpdateDeviceInformation("increment", "number", new NSNumber(3));
Accengage.UpdateDeviceInformation("set", "text", new NSString("foo"));
```

> You can only operate “increment” or “decrement” actions on suitable value NSNumber in iOS

Subscription Tag

Subscription tag is a new API used to mark user interest for a particular topic : public static void SetDeviceTag(string category , string identifier, Dictionary<string, object> parameters)

Android

In Android we are expecting to pass value in the following types :

- Boolean
- Int32
- Int64
- Java.Util.Date
- String

If you try to pass other kind of data, it will be treated as a String

```
Dictionary<string, object> parameters = new Dictionary<string, object>();
parameters["club_league_1"] = "corinthians";
parameters["club_league_2"] = "figueirense";
Accengage.SetDeviceTag("Soccer", "Brazil", parameters);
```

iOS

In iOS we are expecting to pass value in the following types from the Foundation Framework :

- NSDate
- NSNumber
- NSString
- bool

```
Dictionary<string, object> parameters = new Dictionary<string, object>();
parameters.add("club_league_1", new NSString("corinthians"));
parameters.add("club_league_2", new NSString("figueirense"));
Accengage.SetDeviceTag("Soccer", "Brazil", parameters);
```

A device tag can contain up to 5 parameters and all parameter key should be different.

As mentionned with the name of the method, setDeviceTag is an update of the Subscription Tag, it will override then the previous information.

You can set an expiration date for your Subscription Tag by adding the parameter "exp" to the Device Tag where the value is a timestamp for the expiration date :

```
parameters["exp"] = "1543512073";
```

You can unsubscribe a user to a particular topic with the following code :

```
Accengage.DeleteDeviceTag("Soccer", "Brazil");
```

Important
Please note that this feature is only available through Accengage's API

Views

You can identify each views of your application that the SDK can refer to using.

```
TrackScreenDisplay("MyView");
```

where MyView is the name of your view.

iOS
This method must take place in the viewDidAppear and is to be used with TrackScreenDismiss in the viewDidDisappear. Look the Documentation.

DeepLinking

In the Accengage Interface, you can specify some custom parameters linked to click and/or display actions. The Accengage SDK sends a broadcast each time an item is displayed or clicked on in order to give you back these parameters.

Here is how to retrieve your custom parameters from any kind of notification.

Retrieving Inapp/Push custom param

Android

Follow the android documentation available here.

iOS

Follow the iOS documentation available here.

Geofencing

iOS

To enable geofencing and beacon services, call the following methods:

```
AccengageIOS.BMA4SLocationServices.SetGeofenceServiceEnabled(true)
AccengageIOS.BMA4SLocationServices.SetBeaconServiceEnabled(true)
```

Inbox

The inbox feature was made to display some delayed messages to your users. These messages can be a text, a webpage, a richpush, ... with or without buttons.

These messages and buttons interact directly with the Accengage SDK.
You can for instance send a message with an event button, trigger a URL scheme, or display a banner/interstitial when the user opens a message.
This inbox can be used to store push messages received but not opened by the user.
Indeed, with this feature, the user can browse all of the messages you sent to him before.
The inbox can be used as a reminder or as a lite mail client in order to communicate with your clients.

If you want to use this feature, Accengage servers will send all the messages information to your app, but you will need to handle yourself the display part.

With Accengage Inbox you can :

- Obtain messages and display them in your own template
- Update messages status on Accengage Servers (Read/Unread/Archived)

You can find a complete example in our sample project.

After being sure you greatly integrate our sdk with the previous chapter, you can now access to our Inbox Implementation.

Obtain Messages

You will need an async callback to be notified when an Inbox is received. Here is an example of how to define it:

```
Accengage.GetInbox(LoadInbox);

private void LoadInbox(AccengageInbox inbox)
{
    // Messages were downloaded, are ready and contained in a AccengageInbox Object.
}
```

List Messages

Once you retrieved your AccengageInbox object you can access to AccengageInboxMessage. In this example, we will just display a log with the title of each message:

> The process is asynchronous and the callback order is NOT garanteed.
> Double checking the index provided when a message is received is mandatory to avoid errors in your code.

```
private void LoadInbox(AccengageInbox inbox)
{
    // First, we will get each message, so let's do a loop
    for( int i =0; i < inbox.Size; i++){
        inbox.ObtainMessageAtIndex(i, LoadMessage);
    }
}

private void LoadMessage(AccenageInboxMessage message, int position)
{
    // do your treatment
}
```

Message interactions

Now that we have downloaded and displayed our messages, we will allow our user to interact with them.

For example, if you display a message in a list, you may want to provide the user with some interaction if he touches a row.

You will need to give the message the user interacted with to the SDK.

According to the message format, the SDK can either start a predefined action or give it back to you to be displayed.

```
// Call message 'interactWithDisplayHandler' method to hand the message to the SDK
message.InteractWithDisplayHandler(DisplayDetailInbox);

// If this method is called back, I will need to display message content to the user
myself.
private void DisplayDetailInbox(AccengageInboxMessageContent content)
{
    // do your treatment
}
```

Once it is loaded you have fully access to all the content of your message.

Work with Message Buttons

A message content can have some buttons. Here is an example of how to display and interact with these them.

```
private void DisplayDetailInbox(AccengageInboxMessageContent content)
{
    var buttons = messageContent.Buttons;
    if (buttons.Count > 0)
    {
        foreach (var button in buttons)
        {
            // Create ui button
            ...
            uiButton.Click += delegate {
                button.Interact();
            };
        }
    }
}
```

Update message status

You can change the status of a message, here is how to do it:

```
// Marks a message as read or unread
message.Read = true;
message.Read = false;

// Archive or Unarchive a message
message.Archived = true;
message.Archived = false;
```

Other Methods

Restricted Network

You can temporarily restrict the connections in order to tell our SDK to stop flushing network requests.

```
SetNetworkCallsDisabled(true);
```

Disable Accengage Service

If a user want to disable all Trackings, and Sdk Services you can use this code.

```
SetAllServicesDisabled(true);
```