MAUI Plugin Guide for iOS and Android

This guide will walk you through the process of integrating the Plugin.Maui.Apptentive iOS and Android SDKs into your MAUI (Multi-platform App UI) app.

Prerequisites

- Access to your Alchemer Digital account to obtain your App Key and App Signature.
- Must have both an iOS and Android app setup in the dashboard if your Maui app is for both platforms.

Add the Plugin.Maui.Apptentive Package

Before integrating Plugin.Maui.Apptentive into your MAUI app, you need to add the Plugin.Maui.Apptentive package to your project. You can do this by installing the Plugin.Maui.Apptentive NuGet package.

Register Plugin.Maui.Apptentive in your MAUI App

In your `MauiProgram.cs` file, register the Plugin.Maui.Apptentive SDK by adding the following:

```
using Plugin.Maui.Apptentive;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

public static class MauiProgram
{
public static MauiApp CreateMauiApp()
{
var builder = MauiApp.CreateBuilder();

// Create singleton
builder.Services.AddSingleton<IApptentive>(Apptentive.Default);

// Configuration setup and registration

#if __IOS__
var configuration = new Configuration("Your Apptentive iOS App Key", "Your Apptentive iOS App
Signature");
#elif __ANDROID__
var configuration = new Configuration("Your Apptentive Android App Key", "Your Apptentive Android App Sig
nature");
#endif

// Optionally configure logging level and sanitization for debug mode
#if DEBUG
configuration.LogLevel = ApptentiveLogLevel.Verbose;
configuration.ShouldSanitizeLogMessages = false;
#endif

// Register the SDK with the provided configuration
#if __IOS__
Apptentive.Default.Register(configuration, completionHandler);
#elif __ANDROID__
Apptentive.Default.Register(configuration, completionHandler, MainApplication.Current);
#endif



return builder.Build();
}
}
```

Register Main Activity (Android Only)

This step is required for presenting interactions for Android.

In order to display a Prompt or love dialog you must extend the Activity you wish to display it from to adopt IApptentiveActivityInfo, and register the activity in the OnResume method. Be sure to implement using ApptentiveSDK at the top of the MainActivity.cs in YourProject/Platforms/Android.

```
using Android.App;
using Android.Content.PM;
using Android.OS;
using ApptentiveSDK;
using Plugin.Maui.Apptentive;

namespace Plugin.Maui.Apptentive.Sample;
[Activity(Theme = "@style/Maui.SplashTheme", MainLauncher = true, ConfigurationChanges = ConfigChang
es.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout | ConfigCh
anges.SmallestScreenSize | ConfigChanges.Density)]
public class MainActivity : MauiAppCompatActivity, IApptentiveActivityInfo
{
protected override void OnCreate(Bundle savedInstanceState)
{
base.OnCreate(savedInstanceState);
ApptentiveSDK.Apptentive.RegisterApptentiveActivityInfoCallback(this);
}


public Activity ApptentiveActivityInfo {
get {
return this;
}
}

public Activity GetApptentiveActivityInfo() {
return this;
}
}
```

Engaging Events

Allows you to engage with events within your app. It triggers an event with the specified name, which can be used for targeting interactions and tracking user actions

```
Apptentive.Default.Engage("EventName");
```

Presenting Message Center

This method presents the Alchemer Message Center interface within your app. It allows users to view and respond to messages sent by the app owner.

```
Apptentive.Default.PresentMessageCenter();
```

Setting Person Name and Email

If you already know the customer’s email address or name, you can pass them to us to display in the Conversation view on your Alchemer Digital Dashboard.

```
Apptentive.Default.SetPersonName("User Name");
Apptentive.Default.SetPersonEmailAddress("User Email");
```

Add Custom Data

To incorporate custom data for a person, utilize the `addCustomPersonData(key, value)` function. For device-specific custom data, employ the `addCustomDeviceData(key, value)` function. The 'key' parameter denotes the identifier for the custom data, while 'value' represents the actual data to be stored. The supported data types include boolean, string, and numeric values. If you provide new custom data for a key that already exists, the previous data associated with that key will be replaced.

Adding Custom Data

```
ive.Default.addCustomPersonData(key, numericValue);
Apptentive.Default.addCustomDeviceData(key, value);
```

Removing Custom Data

```
Apptentive.Default.removeCustomDeviceData(key);
Apptentive.Default.removeCustomPersonData(key);
```

Multi-User Authentication (iOS Only)

For extra security when handling sensitive data in apps used by multiple people on the same device, consider Customer Authentication. This feature requires app and server modifications to ensure that data is securely handled by the ApptentiveKit SDK. Without Customer Authentication, data remains accessible on unlocked devices, relying solely on iOS safeguards. For step-by-step instructions, check out our Customer Authentication Configuration guide.

Your server will authenticate a customer during login by generating a JSON Web Token (JWT). This token needs specific claims:

- **Subject (sub):** Uniquely identifies the customer.
- **Issuer (iss):** Your company.
- **Issued-at (iat):** Date/time of issue.
- **Expiry (exp):** Set within three days from the issued-at date.

Sign the JWT with your app's JWT Signing Secret found on the API & Development page using the HS-512 algorithm.

Logging a Customer In

After generating a JWT on your server, pass it back to your app for logging in to Alchemer Mobile:

```
Apptentive.Default.LogIn(token, completionHandler);
```

Logging a Customer Out

Ensure to log out a customer whenever their session is invalidated in your app:

```
Apptentive.Default.LogOut()
```

Refreshing Token:

Because of the limited time during which the JWT is valid, it should be periodically refreshed.

```
Apptentive.Default.UpdateToken(token, completionHandler)
```

Message Center Notifications

iOS

Preparing your application for push.

1. Ensure that you have an Entitlements.plist file in the platforms/iOS section of your app. It should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>aps-environment</key>
<string>production</string> //If testing the push notifications in development mode change this to `development`
</dict>
</plist>
```

2. Supply your push certificate in the Alchemer dashboard under the Settings/Integrations/Alchemer Mobile Push tab in .p12 format (If you have a sandbox certificate you can test the push notifications in development mode and with the production push certificate you can test the push notifications in release mode with an ad-hoc provisioning profile).

3. Ensure that your info.plist has `Required background modes` activated with the following values in the array: `App downloads content from the newtork`, `App downloads. Content in response to push notifications`

4. In the app delegate ensure that the Apptentive push notification configurations are added:

```
using Foundation;
using Microsoft.Maui.Storage;
using UIKit;
using UserNotifications;

Namespace App;

[Register("AppDelegate")]
public class AppDelegate : MauiUIApplicationDelegate, IUNUserNotificationCenterDelegate
{
protected override MauiApp CreateMauiApp() => MauiProgram.CreateMauiApp();

public override bool FinishedLaunching(UIApplication application, NSDictionary launchOptions)
{
RegisterRemoteNotifications();
return base.FinishedLaunching(application, launchOptions);
}

private void RegisterRemoteNotifications()
{
if (UIDevice.CurrentDevice.CheckSystemVersion(10, 0))
{
var authOption = UNAuthorizationOptions.Alert | UNAuthorizationOptions.Badge | UNAuthorizationOptions.Sound;
UNUserNotificationCenter.Current.RequestAuthorization(authOption, (granted, error) =>
{
if (granted && error == null)
{
MainThread.BeginInvokeOnMainThread(() =>
{
InvokeOnMainThread(UIApplication.SharedApplication.RegisterForRemoteNotifications);
UNUserNotificationCenter.Current.Delegate = ApptentiveKit.iOS.Apptentive.Shared;
});
}
});
}
}

[Export("application:didReceiveRemoteNotification:fetchCompletionHandler:")]
public void DidReceiveRemoteNotification(UIApplication application, NSDictionary userInfo, Action<UIBackgroundFetchResu
lt> completionHandler)
{
ApptentiveKit.iOS.Apptentive.Shared.DidReceiveRemoteNotification(userInfo, completionHandler);
}

[Export("userNotificationCenter:didReceiveNotificationResponse:withCompletionHandler:")]
public void DidReceiveNotificationResponse(UNUserNotificationCenter center, UNNotificationResponse response, Action co
mpletionHandler)
{
ApptentiveKit.iOS.Apptentive.Shared.DidReceiveNotificationResponse(center, response, completionHandler);
}

[Export("application:didRegisterForRemoteNotificationsWithDeviceToken:")]
public void RegisteredForRemoteNotifications(UIApplication application, NSData deviceToken)
{
ApptentiveKit.iOS.Apptentive.Shared.SetRemoteNotificationDeviceToken(deviceToken);
}
}
```

Android

Push notifications are not yet implemented for Android due to changes in Firebase Cloud Messaging. We are actively migrating to support the new private key configuration for Firebase Cloud Messaging.

Hidden Attachments

Hidden Attachments are messages that you can send from the SDK programmatically, which will be visible to you in the Conversation View, but will not be visible to your customers in Message Center. They are great for sending contextual information or logs from rare crash events.

Sending hidden attachments is possible using the following method:

```
Apptentive.Default.sentAttachment(Text)
```

Custom App Store Rating Dialog (Android Only)

If you want to point users to an app store other than the google play store (like the amazon app store) for reviews, you can set the CustomAppStoreURL property on the configuration object.

Proguard Rules (Android Only)

If you find any errors related to omission of classes here are some proguard rules that may help

```
# WORKING PROGUARD RULES:

-keep class apptentive.com.android.feedback.survey.model.* { *; }

-keep class com.google.android.play.core.review.** { *; }

-keep class com.apptentive.android.sdk.** { *; }

-keep class apptentive.com.android.feedback.model.** { *; }

-keep class apptentive.com.android.feedback.engagement.interactions.InteractionData { *; }

-keep class apptentive.com.android.encryption.** { *; }

-keep class com.google.android.play.review.** { *; }




# Gson library

-keepattributes Signature

-keepattributes *Annotation*
```

```
-dontwarn sun.misc.**

-keep class com.google.gson.** { *; }

-keep class org.apache.commons.io.** { *; }

-keep class org.apache.commons.logging.** { *; }
```

Unsupported Or Upcoming Features

- Multi-User Authentication for Android
- Local customization (for android/iOS) not supported (Android inherits app theme).
- No hidden attachments with image or file data.

Related Articles