

Notification Hub - ANDROID - Full Doc part 2 of 2

BRANCH 8

This is a continuation of Branch 2. (Yes - we've gone back close to the beginning to restart the process for the Android app.)

**Android-Branch-08-
XamarinFormsApp(register-with-Azure-
Notification-Hub)**

First get a registration going on Android's Firebase console:

Follow this tutorial:

<https://docs.microsoft.com/en-us/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gcm>

FIREBASE:

<https://console.firebaseio.google.com/?pli=1>



Add project



Explore a demo project

Project name

 +  + 

XFPushSample 

Tip: Projects span apps across platforms 

Project ID 

xfpushsample 

Country/region 

United States 



Use the default settings for sharing Google Analytics for Firebase data

- ✓ Share your Analytics data with Google to improve Google Products and Services
- ✓ Share your Analytics data with Google to enable technical support
- ✓ Share your Analytics data with Google to enable Benchmarking
- ✓ Share your Analytics data with Google Account Specialists



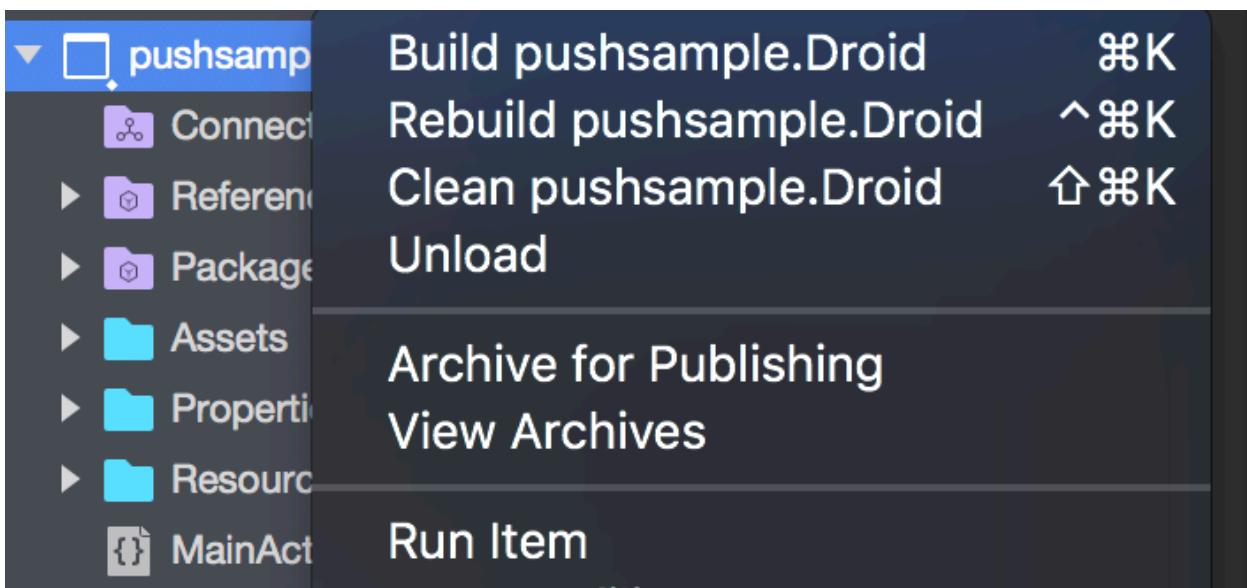
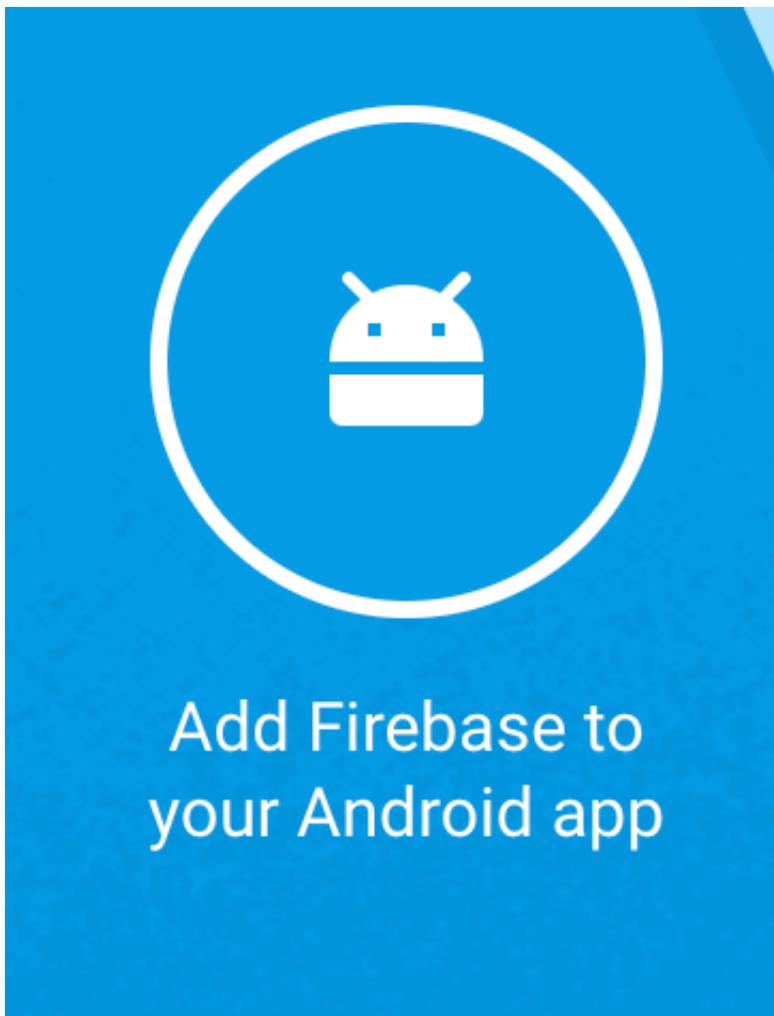
I accept the [controller-controller terms](#). This is required when sharing Analytics data to improve Google Products and Services. [Learn more](#)

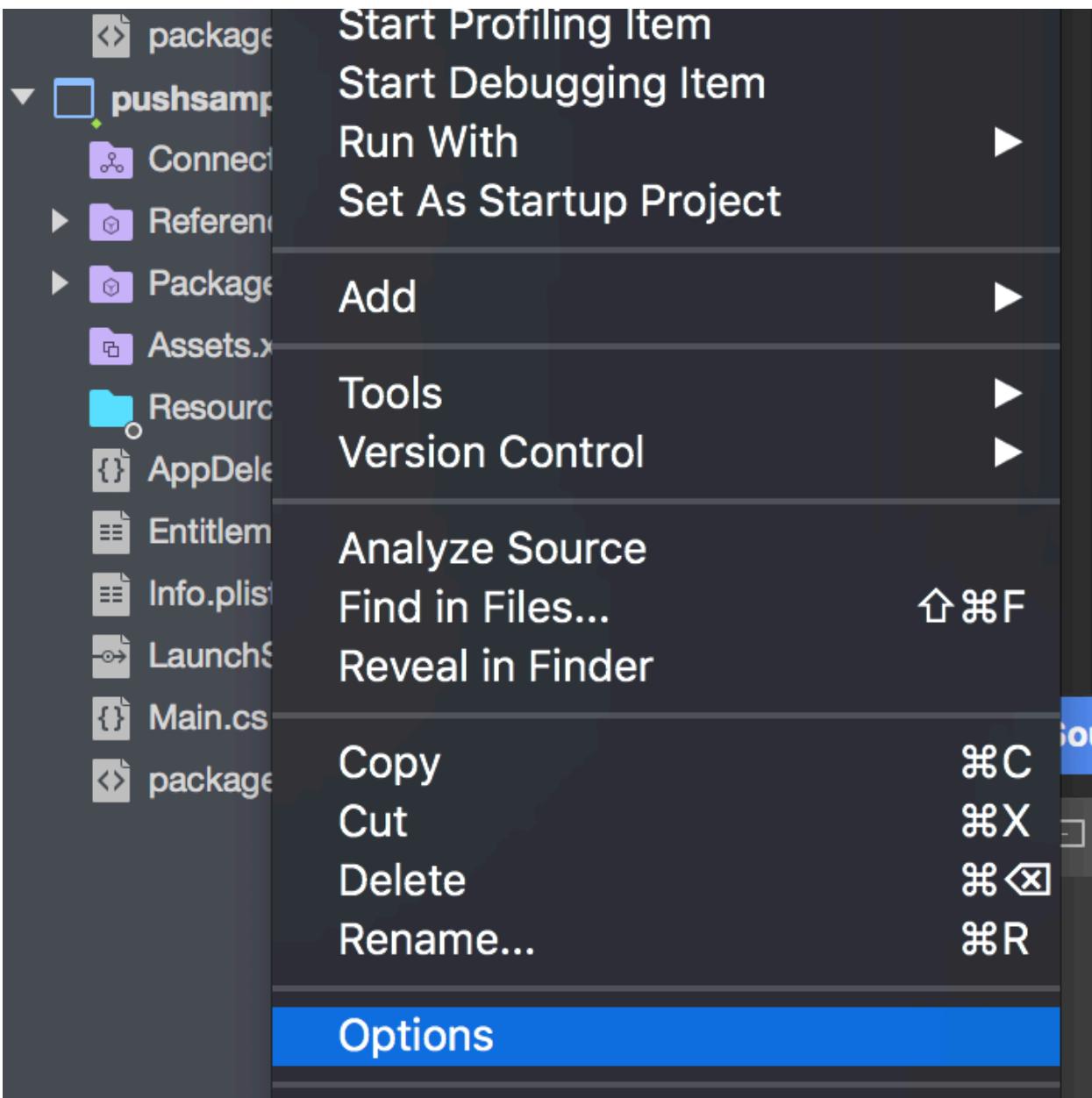


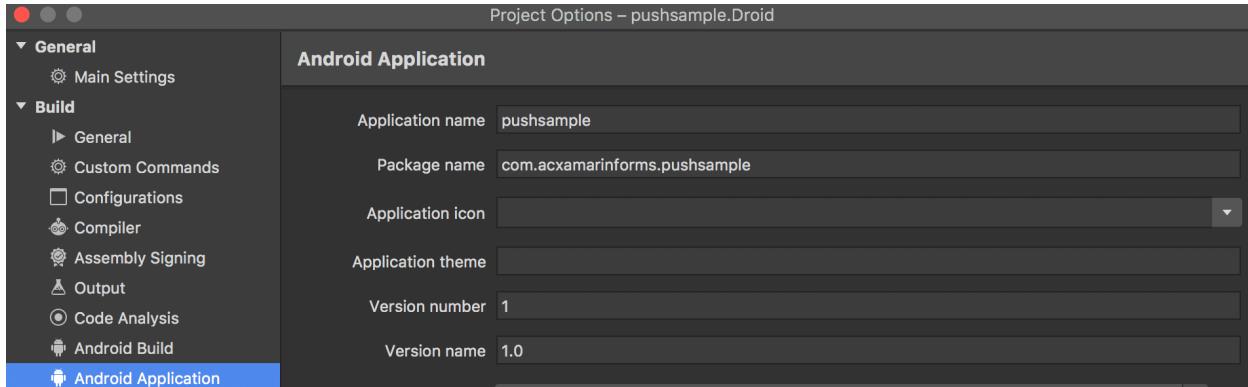
I agree that I am using Firebase services in my app and I agree to the applicable [terms](#).

Cancel

Create project







× Add Firebase to your Android app

1 Register app

Android package name [?](#)

com.acxamarinforms.pushsample

App nickname (optional) [?](#)

pushsample

Debug signing certificate SHA-1 (optional) [?](#)

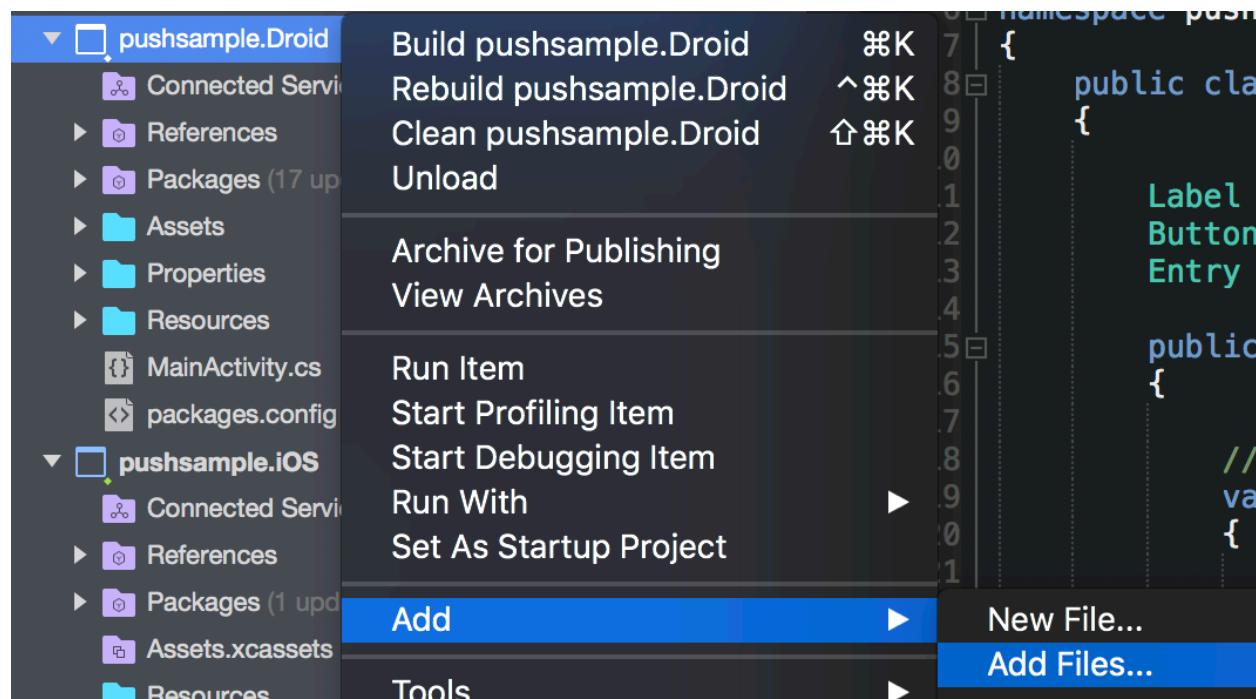
Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

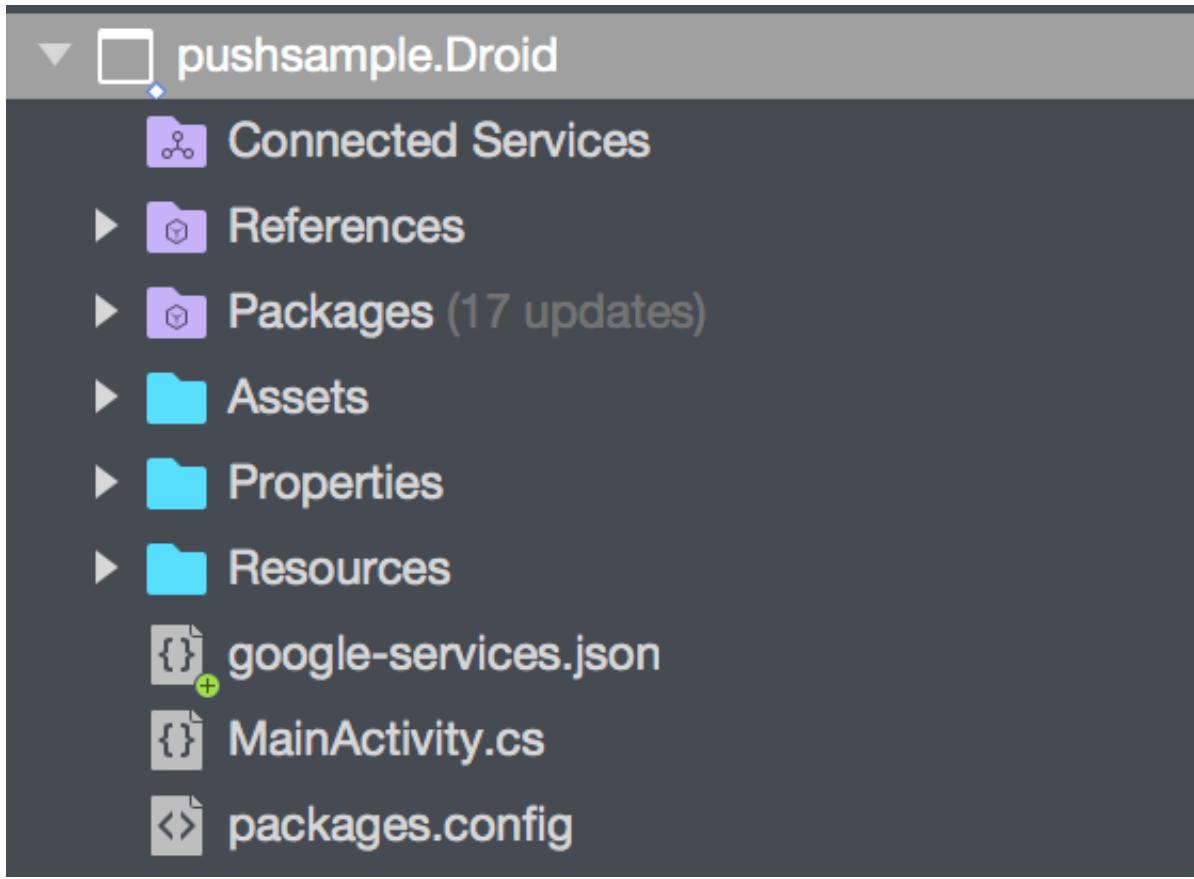
Register app

Download this file:

The screenshot shows a step-by-step guide for adding Firebase to an Android app. Step 1, 'Register app', is completed with a checkmark. Step 2, 'Download config file', is shown with a blue button labeled 'Download google-services.json'. To the right, there are links for 'Instructions for Android Studio below' and 'Unity C++'.

And then add it to your Android project:





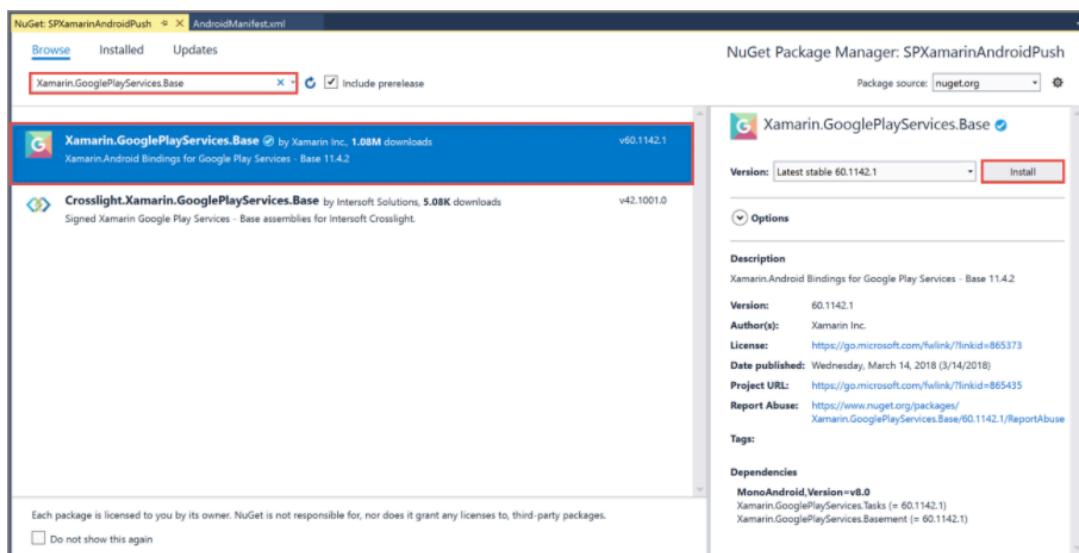
Notice the google-services.json file above.

You're going to need to change the build action of that file to **GoogleServicesJson** (right click on the file and press properties.)

However, you won't see this right away and you'll need add a Nuget package first to see it.

3. Right-click your project, and select
Manage NuGet Packages....

4. Select the **Browse** tab. Search for
Xamarin.GooglePlayServices.Base. Select
Xamarin.GooglePlayServices.Base in the
result list. Then, select **Install**.



And then you might need to CLEAN ALL on your project (Visual Studio - Build > Clean all)

And you may even need to restart Visual Studio.

Now - you'll need to add two more packages:

We'll also need to add these two as well:

5. In the **NuGet Package Manager** window, search for **Xamarin.Firebase.Messaging**. Select **Xamarin.Firebase.Messaging** in the result list. Then, select **Install**.
6. Now, search for **Xamarin.Azure.NotificationHubs.Android**. Select **Xamarin.Azure.NotificationHubs.Android** in the result list. Then, select **Install**.

At this point - you've added the necessary files to get connect to Azure Notification Hubs and Google Firebase Messaging.

However you still haven't added the code necessary to handle the receiving of notifications... we'll handle that first — let's create the actual Azure Notification Hub.

Follow these steps (Under the heading Create a Notification Hub):

<https://docs.microsoft.com/en-us/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gcm#create-a-notification-hub>

Here - I am using a new notification hub for my Android notifications.

Search (Ctrl+I)

Save Discard

«

-  Overview
-  Activity log
-  Access control (IAM)
-  Tags
-  Diagnose and solve problems
-  Quick Start
-  Properties

NOTIFICATION SETTINGS

-  Apple (APNS)
-  Google (GCM)

The API key is the Legacy Server Key found here (Look for the Legacy server key found in the bottom right)

The screenshot shows the Azure portal interface. On the left, the 'Project Overview' sidebar is visible with sections for 'Develop' (Authentication, Database, Storage, Hosting, Functions, ML Kit) and 'Stability'. The main area is titled 'Settings' under 'Cloud Messaging'. It shows tabs for 'General' and 'Cloud Messaging' (which is selected). Below the tabs, there are three sections: 'Project credentials' (with 'Key' selected), 'Server key', and 'Legacy server key' (with a help icon).

Project Overview

Develop

- Authentication
- Database
- Storage
- Hosting
- Functions
- ML Kit

Stability

Settings

Cloud Messaging

Project credentials

Key

Server key

Legacy server key (?)

Now Azure Notification Hubs is setup, and we're going to need to add the code necessary to the Android project to appropriate listen for and handle notifications.

Let's start here:

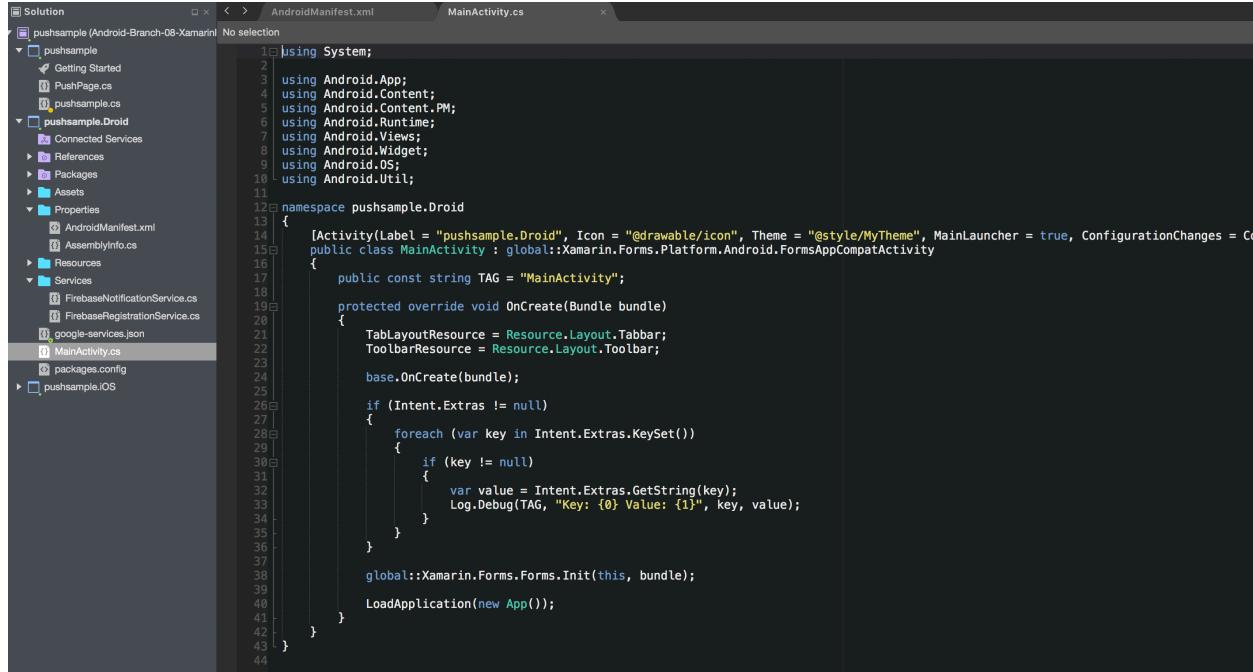
<https://docs.microsoft.com/en-us/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gcm#registering-with-firebase-cloud-messaging>

```

manifest
1 <?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName
3     uses-sdk android:minSdkVersion="26" android:targetSdkVersion="27" />
4<application android:label="pushsample">
5     <receiver android:name="com.google.firebase.iid.FirebaseInstanceIdInternalReceiver" android:exported="fa
6     <receiver android:name="com.google.firebaseio.iid.FirebaseInstanceIdReceiver" android:exported="true" andr
7         <intent-filter>
8             <action android:name="com.google.android.c2dm.intent.RECEIVE" />
9             <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
10            <category android:name="${applicationId}" />
11        </intent-filter>
12    </receiver>
13</application>
14</manifest>

```

Next you'll want to add to your MainActivity.cs



The screenshot shows the Visual Studio IDE interface. On the left is the Solution Explorer pane, which displays the project structure for 'pushsample (Android-Branch-08-Xamarin)'. It includes a 'pushsample' folder containing 'Getting Started', 'PushPage.cs', and 'pushsample.cs'; a 'pushsample.Droid' folder containing 'Connected Services', 'References', 'Assets', 'Properties' (with 'AndroidManifest.xml' and 'AssemblyInfo.cs'), 'Resources', and 'Services' (with 'FirebaseNotificationService.cs' and 'FirebaseRegistrationService.cs'); and an 'google-services.json' file. The 'MainActivity.cs' file is open in the main code editor window. The code in 'MainActivity.cs' is as follows:

```

1 using System;
2 using Android.App;
3 using Android.Content;
4 using Android.Content.PM;
5 using Android.Runtime;
6 using Android.Views;
7 using Android.Widget;
8 using Android.OS;
9 using Android.Util;
10
11 namespace pushsample.Droid
12 {
13     [Activity(Label = "pushsample.Droid", Icon = "@drawable/icon", Theme = "@style/MyTheme", MainLauncher = true, ConfigurationChanges = ConfigurationChanges.ScreenSize | ConfigurationChanges.Orientation)]
14     public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
15     {
16         public const string TAG = "MainActivity";
17
18         protected override void OnCreate(Bundle bundle)
19         {
20             TabLayoutResource = Resource.Layout.Tabbar;
21             ToolbarResource = Resource.Layout.Toolbar;
22
23             base.OnCreate(bundle);
24
25             if (Intent.Extras != null)
26             {
27                 foreach (var key in Intent.Extras.KeySet())
28                 {
29                     if (key != null)
30                     {
31                         var value = Intent.Extras.GetString(key);
32                         Log.Debug(TAG, "Key: {0} Value: {1}", key, value);
33                     }
34                 }
35             }
36
37             global::Xamarin.Forms.Forms.Init(this, bundle);
38
39             LoadApplication(new App());
40         }
41     }
42 }
43
44

```

Let's create a folder called Services - to create the new files: MyFirebasellDService.cs & MyFirebaseMessagingService.cs

MyFirebasellDService.cs

The screenshot shows the Visual Studio IDE interface with the following details:

- Solution Explorer:** Shows the project structure for "pushsample (Android-Branch-08-Xamarin)". It includes the following files and folders:
 - pushsample**: Contains "Getting Started", "PushPage.cs", and "pushsample.cs".
 - pushsample.Droid**: Contains "Connected Services", "References", "Packages", "Assets", "Properties" (with "AndroidManifest.xml" and "AssemblyInfo.cs"), "Resources", and "Services". The "Services" folder contains "MyFirebaseIIDService.cs" and "MyFirebaseMessagingService.cs". It also includes "google-services.json", "MainActivity.cs", and "packages.config".
 - pushsample.iOS**: Represented by a small icon.
- Editor Area:** Displays the code for "MyFirebaseIIDService.cs". The code is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Threading.Tasks;
4 using Android.App;
5 using Android.Util;
6 using Firebase.Iid;
7 using WindowsAzure.Messaging;
8
9 namespace pushsample.Droid.Services
{
    [Service]
    [IntentFilter(new[] { "com.google.firebaseio.INSTANCE_ID_EVENT" })]
    public class MyFirebaseIIDService : FirebaseInstanceIdService
    {
        const string TAG = "MyFirebaseIIDService";
        NotificationHub hub;

        public override void OnTokenRefresh()
        {
            var refreshedToken = FirebaseInstanceId.Instance.Token;
            Log.Debug(TAG, "FCM token: " + refreshedToken);
            Console.WriteLine(TAG, "FCM token: " + refreshedToken);
            SendRegistrationToServer(refreshedToken);
        }

        void SendRegistrationToServer(string token)
        {
            // Register with Notification Hubs
            hub = new NotificationHub(App.NotificationHubName,
                App.ConnectionString, this);

            var tags = new List<string>() { };
            var regID = hub.Register(token, tags.ToArray()).RegistrationId;

            Log.Debug(TAG, $"Successful registration of ID {regID}");
            Console.WriteLine(TAG, $"Successful registration of ID {regID}");
        }
    }
}
```

MyFirebaseMessagingService.cs is the next file we will need:

14. Add the following code to **MyFirebaseMessagingService.cs**:

```
C#  
  
const string TAG = "MyFirebaseMsgService";  
public override void OnMessageReceived(RemoteMessage message)  
{  
    Log.Debug(TAG, "From: " + message.From);  
    if(message.GetNotification() != null)  
    {  
        //These is how most messages will be received  
        Log.Debug(TAG, "Notification Message Body: " + message.GetNotification().Body);  
        SendNotification(message.GetNotification().Body);  
    }  
    else  
    {  
        //Only used for debugging payloads sent from the Azure portal  
        SendNotification(message.Data.Values.First());  
    }  
  
    void SendNotification(string messageBody)  
    {  
        var intent = new Intent(this, typeof(MainActivity));  
        intent.AddFlags(ActivityFlags.ClearTop);  
        var pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntentFlags.OnNewIntent);  
  
        var notificationBuilder = new Notification.Builder(this)  
            .SetContentTitle("FCM Message")  
            .SetSmallIcon(Resource.Drawable.ic_launcher)  
            .SetContentText(messageBody)  
            .SetAutoCancel(true)  
            .SetContentIntent(pendingIntent);  
  
        var notificationManager = NotificationManager.FromContext(this);  
  
        notificationManager.Notify(0, notificationBuilder.Build());  
    }  
}
```

However, it will not work as is.

Oreo (Android 8.0+) has required the use of Notification Channels in your project otherwise those notifications will not send! (Also btw, if you are lacking a small icon you'll also not send those notifications.)

Add the Notification Channel info - and add it to your new NotificationCompat.Builder constructor. It will be ignored if you're running prior to Oreo.

So look at the SendNotification in the code — it will look like this:

```

    public void SendNotification(string messageBody)
    {
        var intent = new Intent(this, typeof(MainActivity));
        intent.AddFlags(ActivityFlags.ClearTop);
        var pendingIntent = PendingIntent.GetActivity(this, 0, intent, PendingIntentFlags.OneShot);

        var notificationManager = NotificationManager.FromContext(this);

        string NotificationChannelIdentifier = "my_channel_101";

        if (VERSION.SdkInt >= Build.VERSION_CODES.O)
        {
            //NotificationChannel channel = new NotificationChannel("notify_001", "Channel human readable title", NotificationManager.Importance);
            NotificationChannel channel = new NotificationChannel(NotificationChannelIdentifier, "Channel human readable title", AndroidNotificationImportance.High);
            channel.EnableLights(true);
            channel.EnableVibration(true);
            channel.SetVibrationPattern(new long[] { 100, 200, 300, 400, 500, 400, 300, 200, 400 });
            channel.Description = "Great description";
            channel.LightColor = Android.Graphics.Color.AliceBlue;
            notificationManager.CreateNotificationChannel(channel);
        }

        var notificationBuilder = new NotificationCompat.Builder(this, NotificationChannelIdentifier)
            .SetSmallIcon(Resource.Drawable.ic_launcher)
            .SetContentTitle("New Notification Message")
            .SetContentText(messageBody)
            .SetContentIntent(pendingIntent)
            .SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification))
            .SetAutoCancel(true);

        notificationManager.Notify(0, notificationBuilder.Build());
        MessagingCenter.Send<object, string>(this, App.NotificationReceivedKey, messageBody);
    }
}

```

BRANCH 9

This is a continuation of Branch 8.

Android-Branch-09- XamarinFormsApp(Tags+Template+Console)

Added tags:

```

    public void SendRegistrationToServer(string token)
    {
        var refreshedToken = FirebaseInstanceId.Instance.Token;
        Log.Debug(TAG, "FCM token: " + refreshedToken);
        Console.WriteLine(TAG, "FCM token: " + refreshedToken);
        SendRegistrationToServer(refreshedToken);
    }

    void SendRegistrationToServer(string token)
    {
        // Register with Notification Hubs
        hub = new NotificationHub(App.NotificationHubName,
            App.ConnectionString, this);

        //var tags = new List<string>() { };
        var tags = new List<string>() { "World", "Politics", "Business", "Technology", "Science", "Sports" };
        var regID = hub.Register(token, tags.ToArray()).RegistrationId;

        Log.Debug(TAG, $"Successful registration of ID {regID}");
        Console.WriteLine(TAG, $"Successful registration of ID {regID}");
    }
}

```

Added Template Registration:

```

43 void SendTemplateRegistrationToServer(string token)
44 {
45     // Register with Notification Hubs
46     hub = new NotificationHub(App.NotificationHubName, App.ConnectionString, this);
47
48     //var tags = new List<string>() { };
49     var tags = new List<string>() { "World" };
50
51     String templateBodyGCM = "{\"data\":{\"message\":\"$(messageParam)\"}}";
52
53     var regID = hub.RegisterTemplate(token, "SimpleGCMTemplate", templateBodyGCM, tags.ToArray()).RegistrationId;
54
55     Log.Debug(TAG, $"Successful registration of ID {regID}");
56     Console.WriteLine(TAG, $"Successful registration of ID {regID}");
57 }
58
59 void SendTemplateMultipleRegistrationToServer(string token)
60 {
61     // Register with Notification Hubs
62     hub = new NotificationHub(App.NotificationHubName, App.ConnectionString, this);
63
64     //var tags = new List<string>() { };
65     var tags = new List<string>() { "World", "Politics", "Business", "Technology", "Science", "Sports" };
66
67     String templateBodyGCM = "{\"data\":{\"message\":\"$(messageParam)\"}}";
68
69     var regID = hub.RegisterTemplate(token, "SimpleGCMTemplate", templateBodyGCM, tags.ToArray()).RegistrationId;
70
71     Log.Debug(TAG, $"Successful registration of ID {regID}");
72     Console.WriteLine(TAG, $"Successful registration of ID {regID}");
73 }

```

Added Console application to send 3 types of notifications:

- 1) Native Notification
- 2) Template Notification
- 3) Template Notification Multiple

NOTE: In case you run into issues - for the console, app - I'm running this App from my Windows Visual Studio

You can switch which notification style you would like to send here:

```

10
11 public static void Main(string[] args)
12 {
13
14     //SendNativeNotificationAsyncNativeAndroid(); //((native registration only)
15     //SendTemplateNotificationAsync(); //((Template registration only)
16     SendTemplateNotificationMultipleAsync(); //Tempalte registration only
17

```

NATIVE MESSAGES:

```

private static async void SendNativeNotificationAsyncNativeAndroid()
{
    //
    //THIS WILL ONLY WORK IF IN YOUR ANDROID APPLICATION - YOU USE A NATIVE REGISTRATION
    //

    NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString
    (
        AzureConstants.AndroidConsoleFullAccessConnectionString,
        AzureConstants.AndroidConsoleApplicationNotificationHubName,
        true
    );
    // Create an array of breaking news categories.
    var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

    var jsonString4 = "{\"data\":{\"message\":\"From your console - native registration\"}}";
    await hubClient.SendGcmNativeNotificationAsync(jsonString4, "World");

    //USE THIS IF YOU HAVE NOT USED TAGS IN YOUR REGISTRATION
    //HOWEVER - THE SAMPLE IN BRANCH 09 + WILL INCLUDE TAGS
    //BE SURE TO DELETE APP FROM PHONE FIRST AND REBUILD TO TRIGGER NEW REGISTRATIONS
    //await hubClient.SendGcmNativeNotificationAsync(jsonString4);
}

```

TEMPLATE MESSAGES:

```

private static async void SendTemplateNotificationAsync()
{
    //
    //THIS WILL ONLY WORK IF IN YOUR IOS APPLICATION - YOU USE A TEMPLATE REGISTRATION
    //

    NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString
    (
        AzureConstants.AndroidConsoleFullAccessConnectionString,
        AzureConstants.AndroidConsoleApplicationNotificationHubName,
        true
    );
    // Create an array of breaking news categories.
    var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

    Dictionary<string, string> templateParams = new Dictionary<string, string>();
    //{
    //    { "messageParam", "Hello World" }
    //};
    //SINGLE NOTIFICATION
    templateParams["messageParam"] = "Breaking " + categories[0] + " News!";
    await hubClient.SendTemplateNotificationAsync(templateParams, "World");

    //MULTIPLE NOTIFICATIONS
    //await hubClient.SendTemplateNotificationAsync(templateParams, categories);
}

```

<https://docs.microsoft.com/en-us/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gem#registering-with-firebase-cloud-messaging>

I'm going to switch it up and add the code from this Xamarin.Forms specific project - however, there will be some modifications as we will not be using the Mobile App back end for our registrations - we will either do it directly from the client or do it via an Azure Function:
<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/push-notifications/azure#android-1>

Notice the process is similar to iOS - you need to (ask for a token) and then (handle the receipt of the token).

In case you're not sure where to put the code for this section: <https://docs.microsoft.com/en->

us/xamarin/xamarin-forms/data-cloud/push-notifications/azure#declaring-the-receiver-in-the-android-manifest: Look below - and then put the code between the <application> tags below:

The screenshot shows the Visual Studio IDE interface. On the left is the Solution Explorer pane displaying the project structure for 'pushsample' (Android-Branch-08-XamarinFormsApp). The 'pushsample.Droid' folder is selected, and its contents are shown in the center pane. The 'pushsample.AndroidManifest.xml' file is open and displayed in the code editor. The XML code is as follows:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:minSdkVersion="15" android:targetSdkVersion="19" android:label="pushsample">
3      <uses-sdk android:minSdkVersion="15" />
4      <application android:label="pushsample">
5          </application>
6      </manifest>
7

```

A closer look at the 'pushsample.AndroidManifest.xml' code in the code editor:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:minSdkVersion="15" android:targetSdkVersion="19" android:label="pushsample">
3      <uses-sdk android:minSdkVersion="15" />
4      <application android:label="pushsample">
5          <receiver android:name="com.google.firebaseio.iid.FirebaseInstanceIdInternalReceiver" android:exported="false" />
6          <receiver android:name="com.google.firebaseio.iid.FirebaseInstanceIdReceiver" android:exported="true" android:label="FirebaseInstanceIDReceiver" />
7              <intent-filter>
8                  <action android:name="com.google.android.c2dm.intent.RECEIVE" />
9                  <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
10                 <category android:name="${applicationId}" />
11             </intent-filter>
12         </receiver>
13     </application>
14 </manifest>
15

```

So there are two things happening here.

First the Firebase Instance Receiver will basically get Firebase Messages (your messages) and Firebase Instance Id (your client registration).

And then it will send those as appropriate to your internal FirebaseInstanceIdInternalReceiver.

ie.

The `FirebaseInstanceIdReceiver` receives `FirebaseInstanceId` and `FirebaseMessaging` events and delivers them to the class that's derived from `FirebaseInstanceIdService`.

For the next step: Implementing the Firebase Instance ID Service
(<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/push-notifications/azure#implementing-the-firebase-instance-id-service>)

You'll need to create a new C# class — and you'll probably also want to put this in a new folder because we'll be adding a couple files to your project.
Create a new folder called: Services.

You'll want these using statements:

```
1 using System;
2 using System.Threading.Tasks;
3 using Android.App;
4 using Android.Util;
5 using Firebase.Iid;
6
```

And you can ignore this line for now - it includes a class we'll make next and also a class we're going to ignore:

```
23
24     // Update notification hub registration
25     Task.Run(async () =>
26     {
27         await AzureNotificationHubService.RegisterAsync(TodoItemManager.DefaultManager.CurrentClient.Get
28     });
29 }
```

For the next step it should look like this - don't worry about the word Push - we'll be fixing / modifying this:

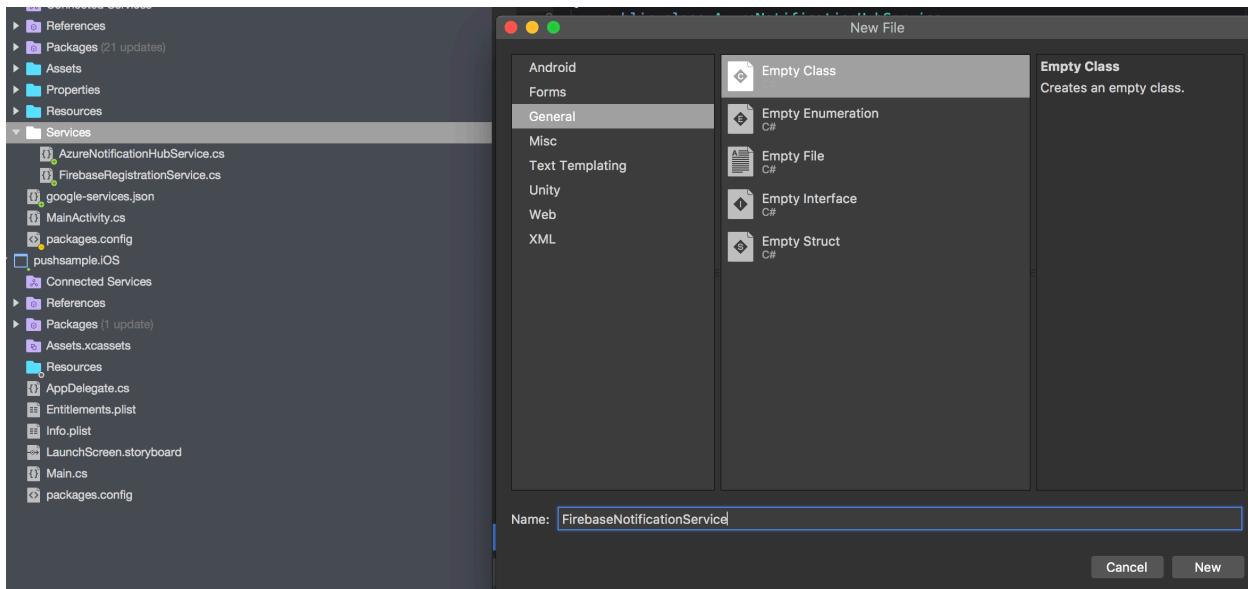
<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/push-notifications/azure#registering-with-the-azure-notification-hub>

```
◆ AzureNotificationHubService ► M RegisterAsync(Push push, string token)
1  using System;
2  using System.Threading.Tasks;
3  using Android.Util;
4  using Newtonsoft.Json.Linq;
5
6  namespace pushsample.Droid.Services
7  {
8      public class AzureNotificationHubService
9      {
10         const string TAG = "AzureNotificationHubService";
11
12         public static async Task RegisterAsync(Push push, string token)
13         {
14             try
15             {
16                 const string templateBody = "{\"data\":{\"message\":\"$(messageParam)\"}}";
17                 JObject templates = new JObject();
18                 templates["genericMessage"] = new JObject
19                 {
20                     {"body", templateBody}
21                 };
22
23                 await push.RegisterAsync(token, templates);
24                 Log.Info("Push Installation Id: ", push.InstallationId.ToString());
25             }
26             catch (Exception ex)
27             {
28                 Log.Error(TAG, "Could not register with Notification Hub: " + ex.Message);
29             }
30         }
31     }
32 }
```

For the next step:

<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/push-notifications/azure#displaying-the-contents-of-a-push-notification>

Create a new class in the Services:



The final file will look like this:

```

FirebaseNotificationService > [+] SendNotification(string messageBody)
1  using System;
2  using Android.App;
3  using Android.Content;
4  using Android.Media;
5  using Android.Support.V4.App;
6  using Android.Util;
7  using Firebase.Messaging;
8
9  namespace pushsample.Droid.Services
10 {
11     [Service]
12     [IntentFilter(new[] { "com.google.firebaseio.MESSAGING_EVENT" })]
13     public class FirebaseNotificationService : FirebaseMessagingService
14     {
15         const string TAG = "FirebaseNotificationService";
16
17         public override void OnMessageReceived(RemoteMessage message)
18         {
19             Log.Debug(TAG, "From: " + message.From);
20
21             // Pull message body out of the template
22             var messageBody = message.Data["message"];
23             if (string.IsNullOrWhiteSpace(messageBody))
24                 return;
25
26             Log.Debug(TAG, "Notification message body: " + messageBody);
27             SendNotification(messageBody);
28         }
29
30         void SendNotification(string messageBody)
31         {
32             var intent = new Intent(this, typeof(MainActivity));
33             intent.AddFlags(ActivityFlags.ClearTop);
34             var pendingIntent = PendingIntent.GetActivity(this, 0, intent, PendingIntentFlags.Once);
35
36             var notificationBuilder = new NotificationCompat.Builder(this)
37                 .SetSmallIcon(Resource.Drawable.ic_stat_ic_notification)
38                 .SetContentTitle("New Todo Item")
39                 .SetContentText(messageBody)
40                 .SetContentIntent(pendingIntent)
41                 .SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification))
42                 .SetAutoCancel(true);
43
44             var notificationManager = NotificationManager.FromContext(this);
45             notificationManager.Notify(0, notificationBuilder.Build());
46         }
47     }
48 }

```

You'll need to add one image to make this fully work:

You'll need this file (Look for the Download button) and then right click and do a save as in the next screen to get it on your computer:

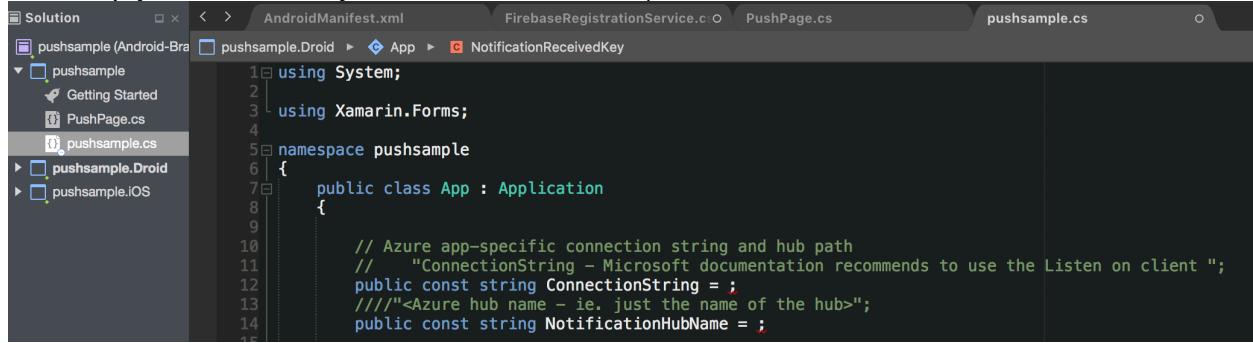
https://github.com/xamarin/xamarin-forms-samples/blob/master/WebServices/TodoAzurePush/Droid/Resources/drawable/ic_stat_ic_notification.png

You'll want to get it in your drawable folder and change the Build Action setting (by right clicking on the file) to an AndroidResource if it is not set that way automatically.

NOW - we're going to switch gears - we're going to now modify the above app to register directly from the app.

<https://docs.microsoft.com/en-us/azure/notification-hubs/xamarin-notification-hubs-push-notifications-android-gcm>

You'll need to create a Notification Hub Client so let's get the appropriate strings from the portal - in the case we want a Access Policy > with only LISTEN privileges. Second, we just the Notification Hub Name (this is similar to steps we've done in previous branches and you can simply use the ones you had before as well.)



```
1 using System;
2 
3 using Xamarin.Forms;
4 
5 namespace pushsample
6 {
7     public class App : Application
8     {
9         // Azure app-specific connection string and hub path
10        // "ConnectionString - Microsoft documentation recommends to use the Listen on client ";
11        public const string ConnectionString = ;
12        //"/>Azure hub name - ie. just the name of the hub";
13        public const string NotificationHubName = ;
14    }
15 }
```

Second - let's do the following:

5. Add an instance variable to **MainActivity.cs** that will be used to show an alert dialog when the app is running:

```
C#  
Copy  
  
public const string TAG = "MainActivity";
```

6. Add the following code to **OnCreate** in **MainActivity.cs** after **base.OnCreate(savedInstanceState)** :

```
C#  
Copy  
  
if (Intent.Extras != null)
{
    foreach (var key in Intent.Extras.KeySet())
    {
        if(key!=null)
        {
            var value = Intent.Extras.GetString(key);
            Log.Debug(TAG, "Key: {0} Value: {1}", key, value);
        }
    }
}
```

Now in Firebase Registration Service - make the following changes:

```

1 using System;
2 using System.Collections.Generic;
3 using System.Threading.Tasks;
4 using Android.App;
5 using Android.Util;
6 using Firebase.Iid;
7 using WindowsAzure.Messaging;
8
9 namespace pushsample.Droid.Services
10 {
11     [Service]
12     [IntentFilter(new[] { "com.google.firebaseio.INSTANCE_ID_EVENT" })]
13     public class FirebaseRegistrationService : FirebaseInstanceIdService
14     {
15         const string TAG = "FirebaseRegistrationService";
16
17         NotificationHub hub;
18
19         public override void OnTokenRefresh()
20         {
21             var refreshedToken = FirebaseInstanceId.Instance.Token;
22             Log.Debug(TAG, "Refreshed token: " + refreshedToken);
23             SendRegistrationTokenToAzureNotificationHub(refreshedToken);
24         }
25
26         void SendRegistrationTokenToAzureNotificationHub(string token)
27         {
28             // Update notification hub registration
29             //Task.Run(async () =>
30             //{
31             //    await AzureNotificationHubService.RegisterAsync(TodoItemManager.DefaultManager.CurrentClient.GetPush(), token);
32             //});
33
34             hub = new NotificationHub(App.NotificationHubName,
35                                     App.ConnectionString, this);
36
37             var tags = new List<string>() { };
38             var regID = hub.Register(token, tags.ToArray()).RegistrationId;
39
40             Log.Debug(TAG, $"Successful registration of ID {regID}");
41
42         }
43     }
44 }

```

BRANCH 10

Continuation of Branch 9:

Android-Branch-10-PushViaFunction-1

Azure Functions will be used to trigger the notification.
The Function is currently HTTP trigger and anonymous.

(THIS IS THE PRINTOUT OF BRANCH 3 EARLIER)

Create Azure Function:

This tutorial will at first use an anonymous function - but security should of course be a consideration in the production environment: <https://docs.microsoft.com/en-us/azure/azure-functions/>

The setup of Azure Function is best described here (in terms of what is required in Visual Studio and a quick setup):

If you've never used an Azure Function - I would start with the following two tutorials.

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function>
<https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-your-first-function-visual-studio>

NAME	PUBLISHER	CATEGORY
 Function App	Microsoft	Web + Mobile 

You can use these setting generally - the Hosting Plan can either be consumption (pay as you go) or App Service plan:

The Consumption plan lets you pay-per-execution and dynamically allocates resources based on your app's load. App Service Plans let you use a predefined capacity allocation with predictable costs and scale.

[Home](#) > [Resource groups](#) > [XamarinFormsNotificationHubSample](#)

Function App

Create

* App name  .azurewebsites.net

* Subscription

* Resource Group 
 Create new Use existing
 ▾

* OS

* Hosting Plan 

Consumption Plan



* Location

South Central US



* Storage 



Create new



Use existing

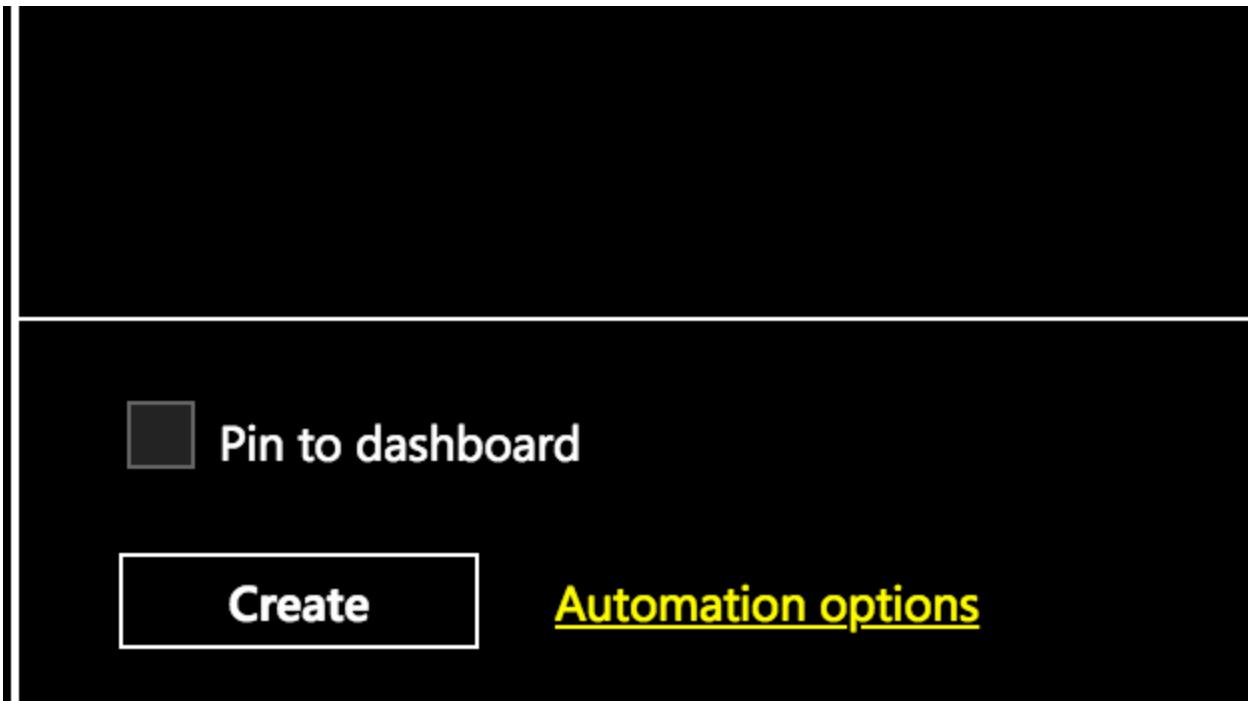
xamarinformsnot92da

Application Insights 

* Application Insights Location 

South Central US





Create a new Functions Project:

The screenshot shows the Microsoft Visual Studio interface with the "New Project" dialog open. The dialog lists various project templates under "Installed" for "Visual C#". The "Azure Functions" template is selected, highlighted with a blue border. The "Name:" field contains "XamarinFormsNotificationsFunctionProject", and the "Location:" field shows the default path "c:\users\lanchun\Source\Repos". A checked checkbox indicates "Create directory for solution". The Visual Studio ribbon is visible at the top, and the Windows taskbar is at the bottom.

Start Page - Microsoft Visual Studio

File Edit View Project Debug Team Tools Test Analyze Window Help

Quick Launch (Ctrl+Q) Andrew Chung AC

Start Page

New Project

.NET Framework 4.6.1 Sort by: Default

Type: Visual C#

Recent

Installed

- Visual C#
 - Windows Universal
 - Windows Classic Desktop
 - Web
 - .NET Core
 - .NET Standard
 - Android
 - Cloud
 - Cross-Platform
 - iOS
 - Test
 - tvOS
 - WCF
 - Visual Basic
 - Visual F#
 - SQL Server
 - Azure Data Lake
 - Stream Analytics
 - Others... Recent Items

Not finding what you are looking for? Open Visual Studio Installer

Name: Location: Solution name:

Browse... Create directory for solution Create new Git repository

OK Cancel

Get Started

Build your first app in 5 minutes

Maximize your productivity in Studio

Take advantage of the new, low-cost and reliable website

Develop modern, fully native

Produce more, fix faster and

Reduce turnaround by making time impact

Recent

This week

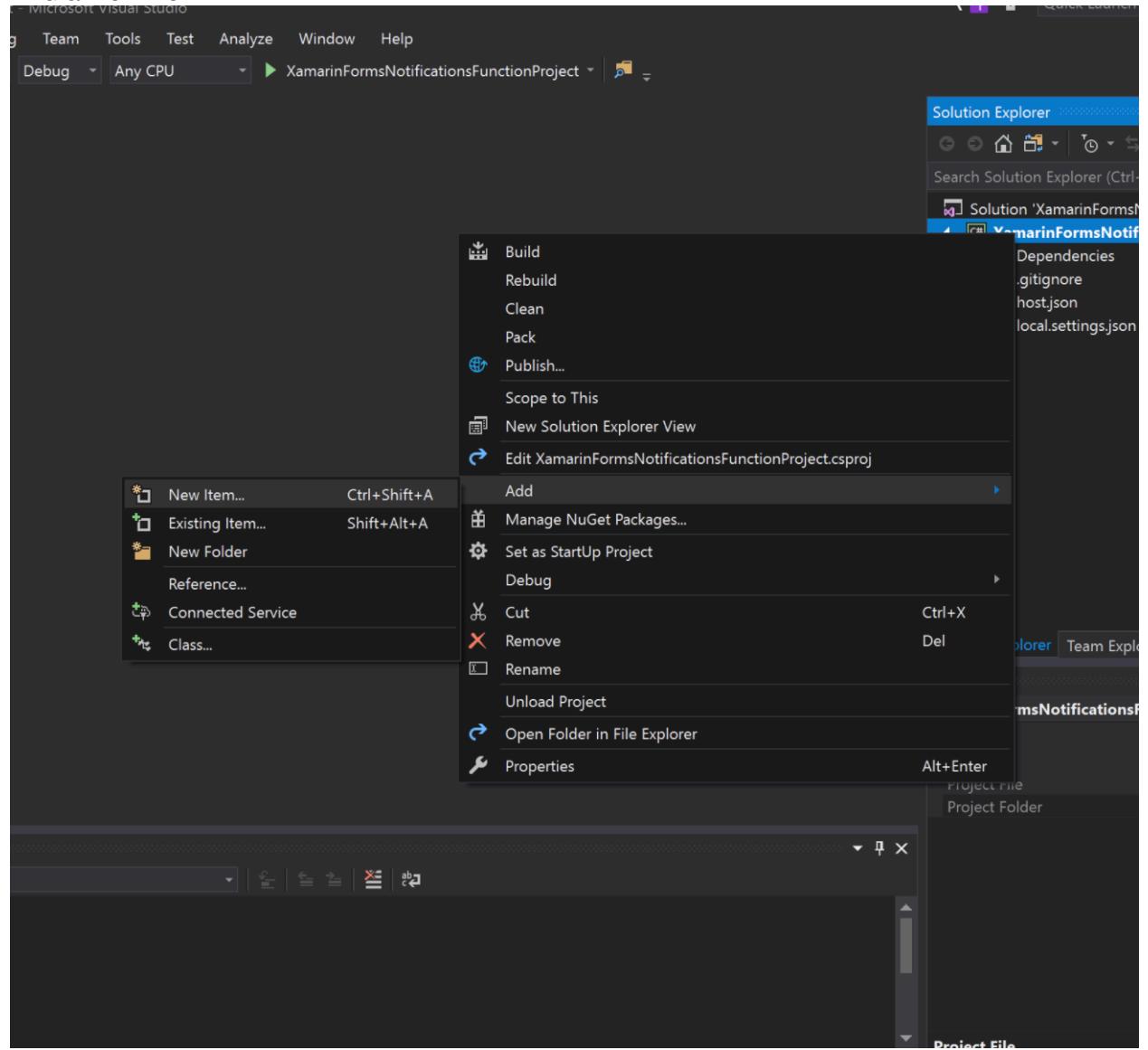
Microsoft.Bot.Sample

XamarinFormsNotificationsFunctionProject

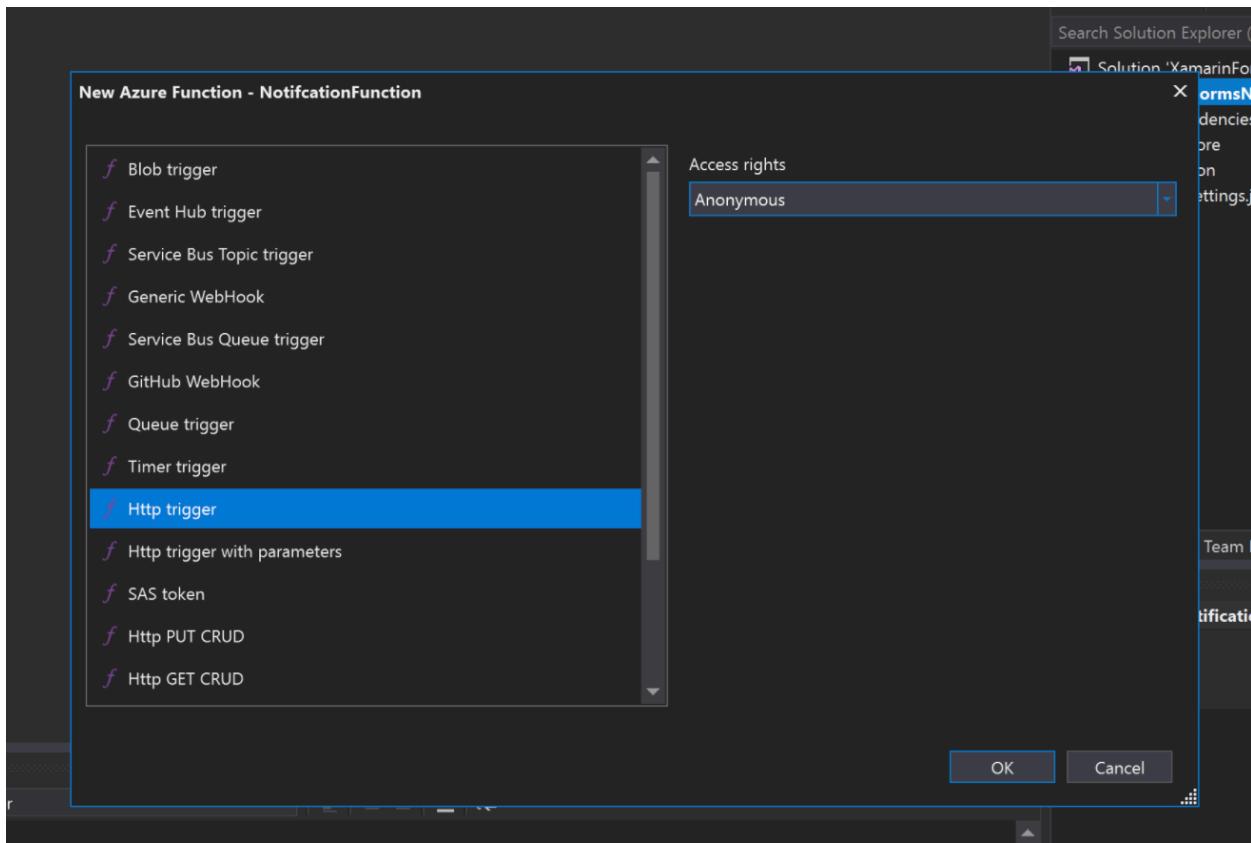
Output Show output from:

9:13 PM 4/19/2018

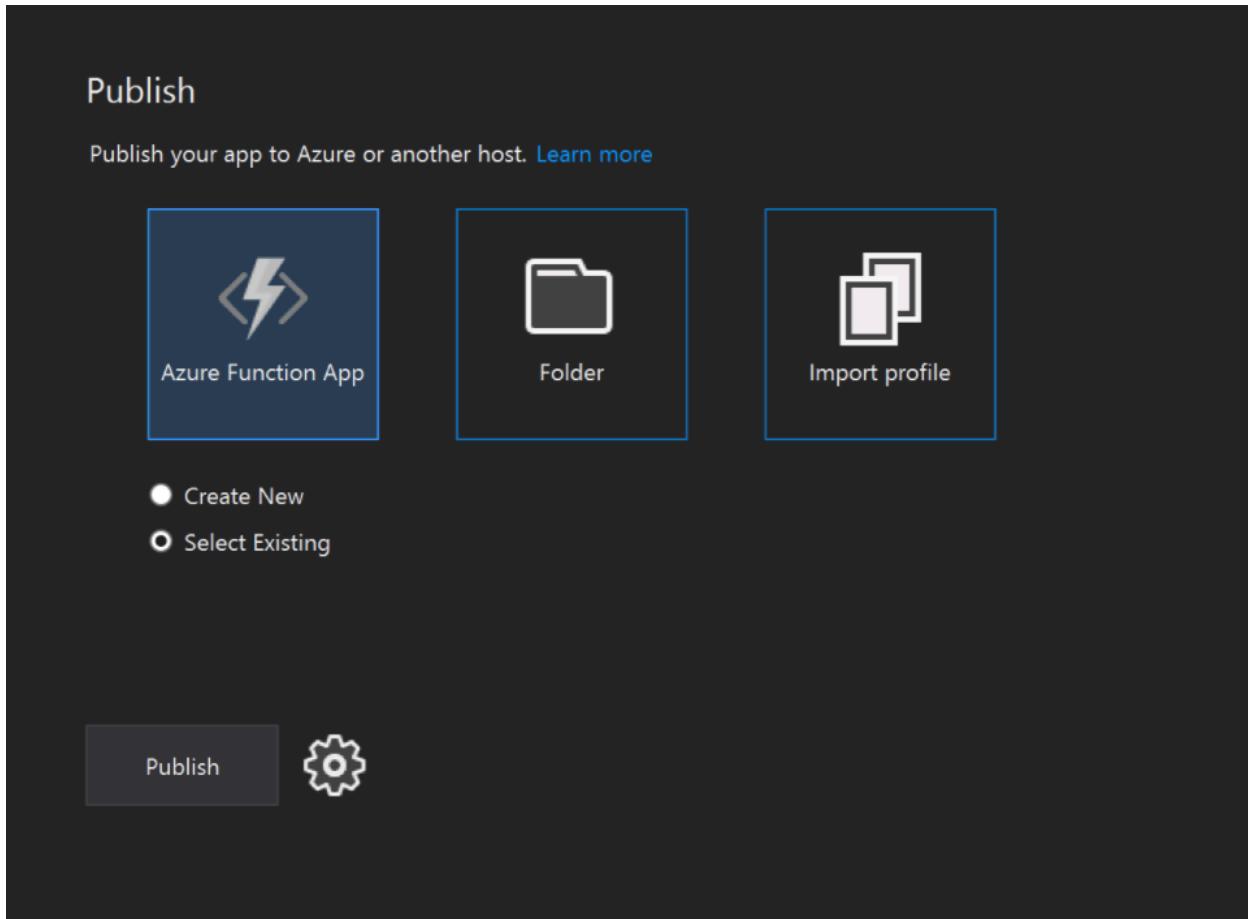
And a New Item:



This tutorial will at first use an anonymous function - but security should of course be a consideration in the production environment: <https://docs.microsoft.com/en-us/azure/azure-functions/>



Right Click your Function project to Publish.
Choose “Select Existing”
Press Publish



You'll be asked for your credentials and then once you log in - you'll be able to see your Resource Group → then Select your Functions App that you created. Then press Publish.

In Visual Studio - you should be able to see the following:

```
Output
Show output from: Build
Adding file (XamarinFormsNotificationFunction\bin\zh-Hant\Microsoft.Data.OData.resources.dll).
Adding file (XamarinFormsNotificationFunction\bin\zh-Hant\Microsoft.Data.Services.Client.resources.dll).
Adding file (XamarinFormsNotificationFunction\bin\zh-Hant\System.Spatial.resources.dll).
Adding file (XamarinFormsNotificationFunction\functionsSdk.out).
Updating file (XamarinFormsNotificationFunction\host.json).
Adding file (XamarinFormsNotificationFunction\NotificationFunction\function.json).
Publish Succeeded.

Restore completed in 29.72 ms for c:\users\anchun\source\repos\xamarinformsnotificationsfunctionproject\xamarinformsnotificationsfunctionproject\xamarinformsn...
```

Go back to the Azure portal - back into the Function app you created:

You should see something like this:

The screenshot shows the Azure Functions Overview page for a function named "XamarinFormsNotificationFunction". The function is listed under "Function Apps" and is currently "Running". Key details include:

- Status:** Running
- Subscription:** Microsoft Azure Internal Consumption
- Resource group:** XamarinFormsNotificationHubSample
- URL:** <https://xamarinformsnotificationfunction.azurewebsites.net>
- Subscription ID:** 4e234d59-b8fc-4ffd-bd01-54591f091d2c
- Location:** South Central US
- App Service plan / pricing tier:** SouthCentralUSPlan (Consumption)

Notice the Url on the upper right?

Try going to that URL on a browser - you should see the following:

The screenshot shows a web browser displaying the Azure Function App landing page. The URL in the address bar is <https://xamarinformsnotificationfunction.azurewebsites.net>. The page content includes:

- The Microsoft Azure logo at the top.
- A large, bold message: "Your Function App is up and running".
- A descriptive text block: "Azure Functions is an event-based serverless compute experience to accelerate your development."
- A "Learn more" button with a right-pointing arrow icon.

Then try <YOUR_URL> / api / {Name of your Function that you defined in your code }

```
namespace XamarinFormsNotificationsFunctionProject
{
    public static class NotificationFunction
    {
        [FunctionName("NotificationFunction")]
        public static async Task<HttpResponseMessage>
    }
}
```

Mine is: <https://xamarinformsnotificationfunction.azurewebsites.net/api/NotificationFunction>

And then it will ask for a name, so try:

[https://xamarinformsnotificationfunction.azurewebsites.net/api/NotificationFunction?
name="Andrew"](https://xamarinformsnotificationfunction.azurewebsites.net/api/NotificationFunction?name=Andrew)

← → ⌂ Secure | [https://xamarinformsnotificationfunction.azurewebsites.net/api/NotificationFunction?name="Andrew"](https://xamarinformsnotificationfunction.azurewebsites.net/api/NotificationFunction?name=Andrew)

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">Hello "Andrew"</string>
```

A small object lesson here!

Notice - I misspelled Notification above — since this very much has a magic string feel to it, please remember to check your spelling quite carefully and don't catch up in small gotchas like this one! (If you're having issues and can't figure it out — consider reviewing URLs and using Postman (<https://www.getpostman.com/>) to double check your work.

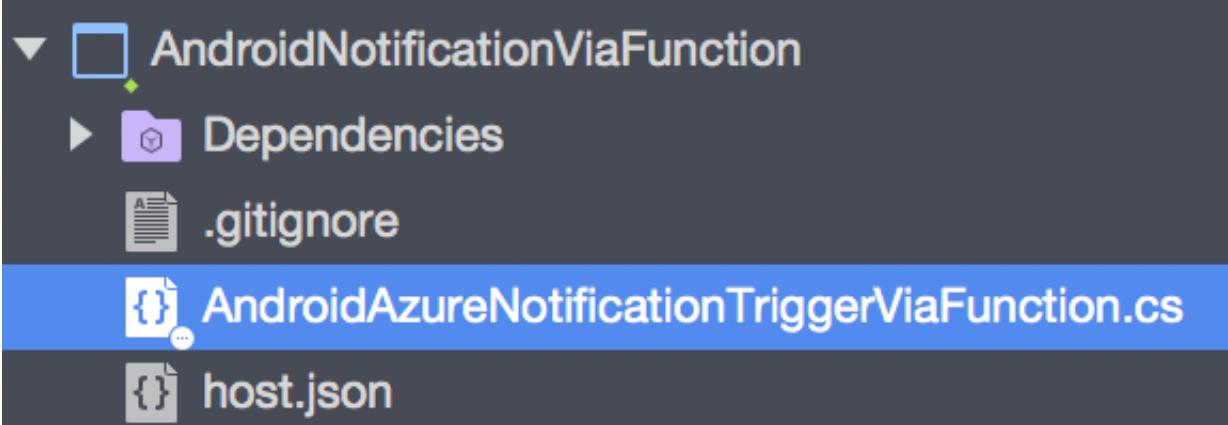
REMEMBER

PHONE (Trigger Native Registration) --> AZURE FUNCTION (Native Registrations) — Pairs With —> Azure Functions (Native Notifications)

OR

PHONE (Trigger Template Registration) --> AZURE FUNCTION (Template Registrations) — Pairs With —> Azure Functions (Template Notifications)

Remember there are three configurations to trigger the Notifications (one is Native the other two are Template)



```
[FunctionName("AndroidNotificationFunctionNative")]
https://YourAzureFunction.net/api/AndroidNotificationFunctionNative
[FunctionName("AndroidNotificationFunctionTemplate")] //sends only 1 notification
https://YourAzureFunction.net/api/AndroidNotificationFunctionTemplate
[FunctionName("AndroidNotificationFunctionTemplateMultiple")] //sends notifications to
several Tags
https://YourAzureFunction.net/api/AndroidNotificationFunctionTemplateMultiple
```

How do you trigger the Notification?

Use a program called: Postman: <https://www.getpostman.com/>
to trigger the URLs:

REMEMBER! You can save and run your Functions locally — you just need to change the first part of the URL (before /api/YourSpecificFunctionName) to the URL shown when you press the |> Button for the Function:

```
AndroidNotificationFunctionNative: http://localhost:7071/api/AndroidNotificationFunctionNative
AndroidNotificationFunctionTemplate: http://localhost:7071/api/AndroidNotificationFunctionTemplate
AndroidNotificationFunctionTemplateMultiple: http://localhost:7071/api/AndroidNotificationFunctionTemplateMultiple
```

But the most important GOTCHA is that you have to PUBLISH your function each time you change it — for it to properly reflect the changes.

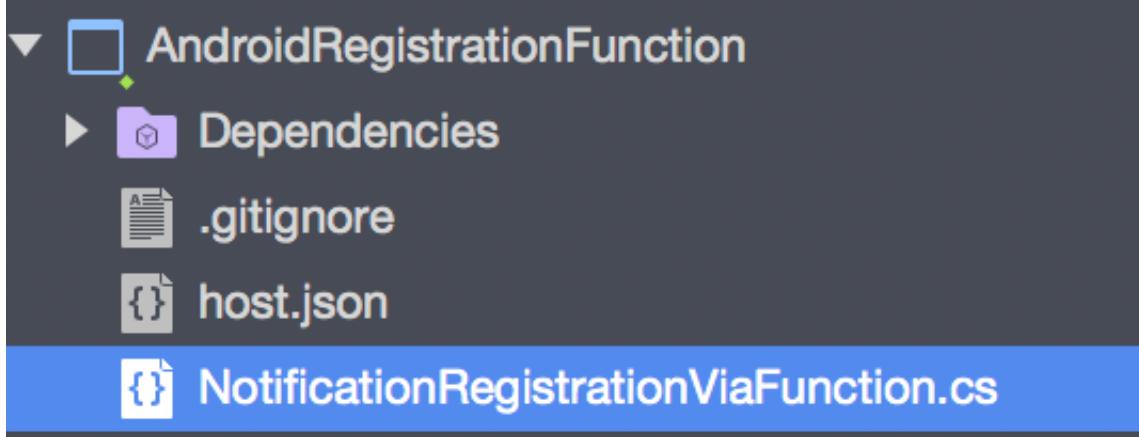
BRANCH 11

Continuation of Branch 10:

Android-Branch-11-Register-Via-Functions-1

REGISTRATIONS:

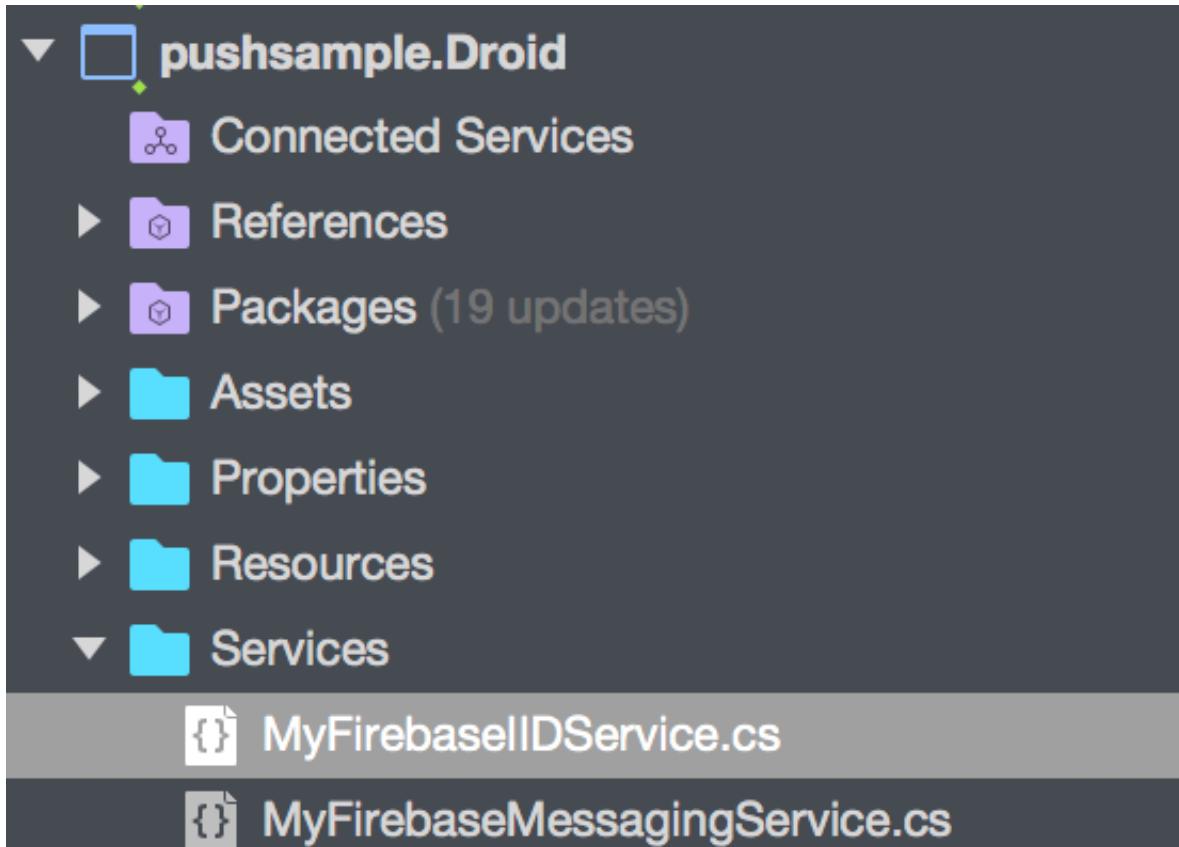
From the Function - there are two configuration to Register the Notifications (Native vs Template):



[FunctionName("NativeAndroidRegistrationWithTagsAndUsername")]
<https://YourAzureFunction.net/api/GetNativeAndroidRegistrationIdPassingHandleUsername/{handleString}>

[FunctionName("TemplateAndroidRegistrationWithTagsUsername")]
<https://YourAzureFunction.net/api/GetTemplateAndroidRegistrationWithTagsUsername/{handleString}>

TRIGGERING / TOGLGING REGISTRATION TYPE FROM THE PHONE:
You've defined the registration - you need to now trigger it from the App.



TOGGLE BETWEEN THE COMMENTED OUT LINES OF CODE:

```
32
33     public override void OnTokenRefresh()
34     {
35         var refreshedToken = FirebaseInstanceId.Instance.Token;
36         Log.Debug(TAG, "FCM token: " + refreshedToken);
37         Console.WriteLine(TAG, "FCM token: " + refreshedToken);
38
39         //REGISTRATION FROM APP DIRECTLY
40         //SendNativeRegistrationToServer(refreshedToken);
41         //SendTemplateRegistrationToServer(refreshedToken);
42         //SendTemplateMultipleRegistrationToServer(refreshedToken);
43
44         //FUNCTION REGISTRATION
45         //NativeAndroidSendRegistrationToServerAsync(refreshedToken);
46         TemplateAndroidSendRegistrationToServerAsyncTemplate(refreshedToken);
47
48         //FUNCTION REGISTRATION WITH USERNAME
49         //NativeAndroidSendRegistrationToServerAsyncWithClientUsername(refreshedToken);
50         //TemplateAndroidSendRegistrationToServerAsyncTemplateWithClientUsername(refreshedToken);
51
52 }
```

SIDE NOTE - AS A DEBUGGING STEP:

```

1 ↴ {
2   "platform": "gcm",
3   "handle": "SCRAMBLED_TAG_EGqmhkQVNzVUzc4cp9oCnbH2UnnxSRfg2pNP8UoeL0pGTAf6nuFWNhxi6ifvtFjGokYsfas3roj2jro23fo23m3ofm2om3f3UQa",
4   "tags": [ "Functions", "World", "Politics", "Business", "Technology", "Science", "Sports" ],
5   "name": "hi",
6   "jsonBodyTemplates": "{\"data\":{\"message\":\"${messageParam}\",\"}}",
7   "expiryTemplate": "0"
8 }
9

```

By the way, here's what I was using in my in my Postman Posts.

```
{
  "platform": "gcm",
  "handle": "SCRAMBLED_TAG_EGqmhkQVNzVUzc4cp9oCnbH2UnnxSRfg2pNP8UoeL0pGTAf6nuFWNhxi6ifvtFjGokYsfas3roj2jro23fo23m3ofm2om3f3UQa",
  "tags": [ "Functions", "World", "Politics", "Business", "Technology", "Science", "Sports" ],
  "name": "hi",
  "jsonBodyTemplates": "{\"data\":{\"message\":\"${messageParam}\",\"}}",
  "expiryTemplate": "0"
}
```

BRANCH 12

This is a continuation of Branch 11.

Android-Branch-12-Register-and-Trigger-Username

The only difference is that there is a tag → to distinguish it from other “tags” we’re going to follow a pattern where we create a prefix to the tag (username:). The tag will look like this “username:YourUsername”.

This example will add the tags in various various parts of the code — and adds it to the array of tags at the time of registration.

Here's an example of adding the username at the time of Registrations.



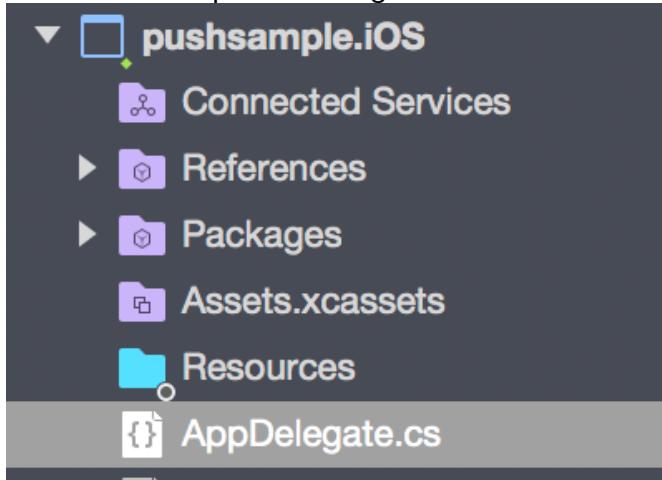
NATIVE

```
113
114           //ADD CHECK IF USER IS ALLOWED TO USE THESE TAGS
115           registration.Tags = new HashSet<string>(deviceUpdate.Tags);
116           var androidNativeUsername = "AndroidFriendlyUser101";
117           registration.Tags.Add("username:" + androidNativeUsername);
118
```

TEMPLATE

```
163
164           var stringTags = deviceRegistrationObjectWithTemplate.Tags;
165
166           Array.Resize(ref stringTags, stringTags.Length + 1);
167           var AndroidTemplateUsername = "AndroidTemplateUser101";
168           stringTags[stringTags.Length - 1] = "username:" + AndroidTemplateUsername;
169
```

Here's an example of adding the username from the client (phone):



NATIVE

```
253
254           string[] stringTags = new string[] { "Functions", "World", "Politics", "Business", "Technology", "Science", "Sports" };
255
256           Array.Resize(ref stringTags, stringTags.Length + 1);
257           var AndroidClientUsername = "AndroidClientNewUser101";
258           stringTags[stringTags.Length - 1] = "username:" + AndroidClientUsername;
259
```

TEMPLATE

```

314
315     string[] stringTags = new string[] { "Functions", "World", "Politics", "Business", "Technology", "Science", "Sports" };
316
317     Array.Resize(ref stringTags, stringTags.Length + 1);
318     var AndroidClientUsername = "AndroidClientNewUser101";
319     stringTags[stringTags.Length - 1] = "username:" + AndroidClientUsername;
320

```

Here's the Triggering of the Notifications with Username - simply add the Username with the addition of the prefix:

CONSOLE APPLICATION:

Here we used a simplistic send - Branch 6 (iOS) shows a slightly more complicated customization of sent messages:

Native:

```

private static async void SendNativeNotificationAsyncNativeAndroidUsername()
{
    //
    //THIS WILL ONLY WORK IF IN YOUR ANDROID APPLICATION - YOU USE A NATIVE REGISTRATION
    //

    NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString
    (
        AzureConstants.AndroidConsoleFullAccessConnectionString,
        AzureConstants.AndroidConsoleApplicationNotificationHubName,
        true
    );
    // Create an array of breaking news categories.
    var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

    Array.Resize(ref categories, categories.Length + 1);
    var AndroidClientUsername = "AndroidNativeUser101";
    categories[categories.Length - 1] = "username:" + AndroidClientUsername;

    var jsonString4 = "{\"data\":{\"message\":\"From your console - native registration\"}}";
    await hubClient.SendGcmNativeNotificationAsync(jsonString4, "World");

    //USE THIS IF YOU HAVE NOT USED TAGS IN YOUR REGISTRATION
    //HOWEVER - THE SAMPLE IN BRANCH 09 + WILL INCLUDE TAGS
    //BE SURE TO DELETE APP FROM PHONE FIRST AND REBUILD TO TRIGGER NEW REGISTRATIONS
    //await hubClient.SendGcmNativeNotificationAsync(jsonString4);
}

```

Template:

```

private static async void SendTemplateNotificationAsyncUsername()
{
    // THIS WILL ONLY WORK IF IN YOUR IOS APPLICATION - YOU USE A TEMPLATE REGISTRATION
    //

    NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString
    (
        AzureConstants.AndroidConsoleFullAccessConnectionString,
        AzureConstants.AndroidConsoleApplicationNotificationHubName,
        true
    );
    // Create an array of breaking news categories.
    var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

    Array.Resize(ref categories, categories.Length + 1);
    var AndroidClientUsername = "AndroidTemplateUser101";
    categories[categories.Length - 1] = "username:" + AndroidClientUsername;

    Dictionary<string, string> templateParams = new Dictionary<string, string>();
    //{
    //    { "messageParam", "Hello World" }
    //};
    //SINGLE NOTIFICATION
    templateParams["messageParam"] = "Breaking " + categories[0] + " News!";
    await hubClient.SendTemplateNotificationAsync(templateParams, "World");

    //MULTIPLE NOTIFICATIONS
    //await hubClient.SendTemplateNotificationAsync(templateParams, categories);
}

```

Template Multiple:

```

private static async void SendTemplateNotificationMultipleAsyncUsername()
{
    // THIS WILL ONLY WORK IF IN YOUR IOS APPLICATION - YOU USE A TEMPLATE REGISTRATION
    //

    NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString
    (
        AzureConstants.AndroidConsoleFullAccessConnectionString,
        AzureConstants.AndroidConsoleApplicationNotificationHubName,
        true
    );
    // Create an array of breaking news categories.
    var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };
    |
    Array.Resize(ref categories, categories.Length + 1);
    var AndroidClientUsername = "AndroidTemplateUser101";
    categories[categories.Length - 1] = "username:" + AndroidClientUsername;

    Dictionary<string, string> templateParams = new Dictionary<string, string>();
    //{
    //    { "messageParam", "Hello World" }
    //};
    templateParams["messageParam"] = "Breaking News!";

    foreach (var category in categories)
    {
        templateParams["messageParam"] = "Breaking " + category + " News!";
        await hubClient.SendTemplateNotificationAsync(templateParams, categories);
    }
}

```

Here's the Triggering of the Notifications with Username - simply add the Username with the addition of the prefix:

FUNCTION APPLICATION:

Native:

```

[FunctionName("AndroidNotificationFunctionNative")]
public static async Task<HttpResponseMessage> RunAndroidNotificationFunctionNative([HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = null)]HttpRequest
{
    // THIS WILL ONLY WORK IF IN YOUR ANDROID APPLICATION - YOU USE A NATIVE REGISTRATION
    //

    try
    {
        NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString
        (
            AndroidAzureConstants.AndroidAzureConstants.AndroidConsoleFullAccessConnectionString,
            AndroidAzureConstants.AndroidAzureConstants.AndroidConsoleApplicationNotificationHubName,
            true
        );
        // Create an array of breaking news categories.
        var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

        Array.Resize(ref categories, categories.Length + 1);
        var nativeUsername = "AndroidNativeUser101";
        categories[categories.Length - 1] = "username:" + nativeUsername;

        for (int i = 0; i < categories.Length; i++)
        {
            //FOR FORMAT REQUIRING MODIFIED MESSAGES IN YOUR ALERT
            var _data = new Data()
            {
                message = String.Format("From Azure Functions - {0} native registration.", categories[i])
            };

            var _rootObject = new RootObject
            {
                data = _data
            };

            string jsonString = JsonConvert.SerializeObject(_rootObject);

            await hubClient.SendGcmNativeNotificationAsync(jsonString, categories[i]);
        }

        //FOR FORMATS NOT REQUIRING MODIFIED MESSAGES IN YOUR ALERT
        //string jsonString2 = String.Format("{\"aps\":{\"alert\":\"From your console - native registration\"}}");
        //await hubClient.SendAppleNativeNotificationAsync(jsonString2, categories[i]);
    }

    return req.CreateResponse(System.Net.HttpStatusCode.OK, "Success - Notification Function triggered - Native");
}

catch (Exception exception)
{
    Debug.WriteLine("Error: ", exception.Message);
}

return req.CreateResponse(System.Net.HttpStatusCode.BadRequest, "Bad Request - Notification Function triggered - Native");

```

It was necessary to create an object that would serialize and mimic the Json string that follows the format of notification that Apple expects from your notification message. Why was this done? This was necessary to create text in the messages that would accept “variables” that could be customized to change the wording of the notification. In the above example you’ll see it as categories[i].

Template:

```

[FunctionName("AndroidNotificationFunctionTemplate")]
public static async Task<HttpResponseMessage> RunAndroidNotificationFunctionTemplate([HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = null)]HttpTrigger)
{
    // THIS WILL ONLY WORK IF IN YOUR ANDROID APPLICATION - YOU USE A TEMPLATE REGISTRATION
    //

    try
    {
        NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString(
            (
                AndroidAzureConstants.AndroidAzureConstants.AndroidConsoleFullAccessConnectionString,
                AndroidAzureConstants.AndroidAzureConstants.AndroidConsoleApplicationNotificationHubName,
                true
            )
        );
        // Create an array of breaking news categories.
        var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

        Array.Resize(ref categories, categories.Length + 1);
        var nativeUsername = "AndroidTemplateUser101";
        categories[categories.Length - 1] = "username:" + nativeUsername;

        Dictionary<string, string> templateParams = new Dictionary<string, string>();
        //{
        //    { "messageParam", "Hello World" }
        //};
        //SINGLE NOTIFICATION
        templateParams["messageParam"] = "Breaking " + categories[0] + " News!";
        await hubClient.SendTemplateNotificationAsync(templateParams, "World");

        //MULTIPLE NOTIFICATIONS
        //await hubClient.SendTemplateNotificationAsync(templateParams, categories);

        return req.CreateResponse(System.Net.HttpStatusCode.OK, "Success - Notification Function triggered - Template");
    }
    catch (Exception exception)
    {
        Debug.WriteLine("Error: ", exception.Message);
    }

    return req.CreateResponse(System.Net.HttpStatusCode.BadRequest, "Bad Request - Notification Function triggered - Template");
}

```

Template Multiple:

```

[FunctionName("AndroidNotificationFunctionTemplateMultiple")]
public static async Task<HttpResponseMessage> RunAndroidNotificationFunctionTemplateMultiple([HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route =
{
    // THIS WILL ONLY WORK IF IN YOUR ANDROID APPLICATION - YOU USE A TEMPLATE REGISTRATION
    //
    try
    {
        NotificationHubClient hubClient = Microsoft.Azure.NotificationHubs.NotificationHubClient.CreateClientFromConnectionString(
            (
                AndroidAzureConstants.AndroidAzureConstants.AndroidConsoleFullAccessConnectionString,
                AndroidAzureConstants.AndroidAzureConstants.AndroidConsoleApplicationNotificationHubName,
                true
            )
        );
        // Create an array of breaking news categories.
        var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports" };

        Array.Resize(ref categories, categories.Length + 1);
        var nativeUsername = "AndroidTemplateUser101";
        categories[categories.Length - 1] = "username:" + nativeUsername;

        Dictionary<string, string> templateParams = new Dictionary<string, string>();

        foreach (var category in categories)
        {
            templateParams["messageParam"] = "Breaking " + category + " News!";
            await hubClient.SendTemplateNotificationAsync(templateParams, categories);
        }

        return req.CreateResponse(System.Net.HttpStatusCode.OK, "Success - Notification Function triggered - Template Multiple");
    }
    catch (Exception exception)
    {
        Debug.WriteLine("Error: ", exception.Message);
    }

    return req.CreateResponse(System.Net.HttpStatusCode.BadRequest, "Bad Request - Notification Function triggered - Template Multiple");
}

```

NOTE: If you need to add the tags later on (say - a week or after the registration) — you can either CreateOrUpdate the Native Registration or you can simply register the Template Registration.

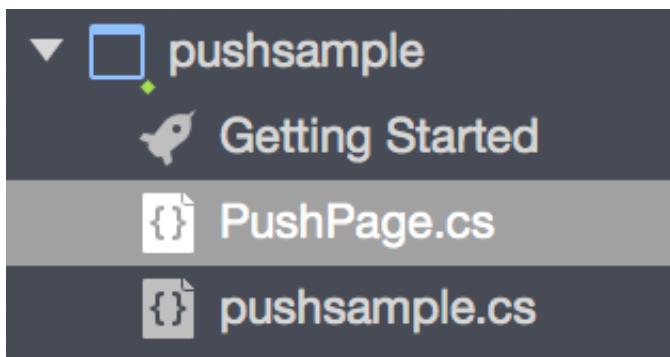
BRANCH 14

This is a continuation of Branch 12.

Android-Branch-14-TriggerButtonPress

In this example, we're simply using Buttons to trigger notifications.

You can press the button and then quickly close the app if you want to trigger notifications and see how they operate with the app closed.



The Function is currently HTTP trigger and anonymous.

So all we do, is set up a HttpClient - configure it with the current URL to trigger the function.

This is one example of the format of the buttons - there is one for each type of Notification (Native / Template / MultipleTemplate)

```
async void OnBtnSendClicked_MultipleTemplate(object sender, EventArgs e)
{
    Debug.WriteLine($"Sending message notification MultipleTemplate to URI {AzureNotificationsViaFunctionsURLS.AndroidMultipleTemplateApiURL}");

    var httpRequest = new HttpRequestMessage
    {
        Method = new HttpMethod("POST"),
        RequestUri = new Uri(AzureNotificationsViaFunctionsURLS.AndroidMultipleTemplateApiURL),
    };

    var result = await _httpClient.SendAsync(httpRequest).ConfigureAwait(false);
    Debug.WriteLine("Send result: " + result.IsSuccessStatusCode);
}
```

Remember:

REMEMBER

PHONE (Trigger Native Registration) --> AZURE FUNCTION (Native Registrations) — Pairs With —> Azure Functions (Native Notifications)

OR

PHONE (Trigger Template Registration) --> AZURE FUNCTION (Template Registrations) —
Pairs With —> Azure Functions (Template Notifications)

See more at Branch 5 on how to change the registration of the application.