**Azure Notification Hub**

**Environment set up for Android development using React-Native**

#### Install Android Studio

#### Create or Set up Emulator in Android studio

#### Install Android SDK.

#### Configure the ANDROID\_HOME environment variable

#### Add platform-tools to Path

#### For reference follow the link below and use React Native CLI Quick start tab

#### <https://reactnative.dev/docs/getting-started>

#### Steps creating Android application from Android Studio

#### 

## Create an Android Studio Project

1. Launch Android Studio.
2. Select **File**, point to **New**, and then select **New Project**.
3. On the **Choose your project** page, select **Empty Activity**, and then select **Next**.
4. On the **Configure your project** page, take the following steps:
   1. Enter a name for the application.
   2. Specify a location in which to save the project files.
   3. Select **Finish**.

#### 

#### Configuring the created Project to Firebase Cloud Messaging(FCM)

#### Create Firebase Console Account using your Gmail account

#### URL: <https://console.firebase.google.com/>

#### Then login into Firebase console and create Project in firebase console

#### Once we login to console we can see below screenshot

#### 

#### Click on Add Project browser will navigate to below screen, where we enter project Name( e.g. PushNotificationsApp) and click Continue button

#### 

#### Then browser will navigate to below screen, where we will select Google Analytics account as Default Account for firebase as below screenshot and click create Project button that create a project and will be displayed on Home page of firebase console

#### 

#### Add the created Android application to the project created in Firebase console by following below steps:

#### Click on the android symbol in your project in firebase console as shown in below screen shot

#### Then browser will navigate to below details screen as shown below

#### 

#### Get Application Id from App level build.gradel file from android prj for Android package field and then click on Register button.

#### Then download google-service.json file and place this json in app folder of android project

#### Make following changes to project level build.gradel file to Add Firebase SDK as shown below

#### 

#### And make following changes to App level build.gradel file

#### 

#### Then open project in Android studio and In File menu select Sync project with Gradel Files. Which adds firebase SDK to project and click next button in firebase console.

#### Run the android application and verify the installation or skip this step by clicking Skip button.

#### Now our android app is ready with firebase SDK.

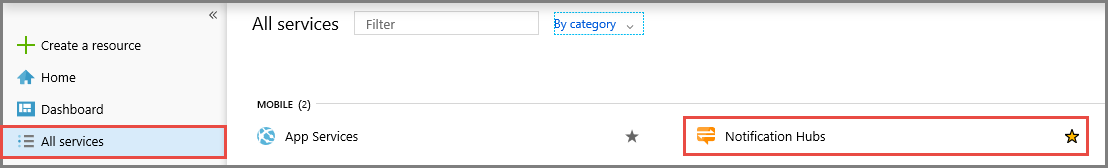
#### Then go to project settings of the project we created and select cloud messaging tab and copy server key. We will be using this key for configuring this to azure notification Hub.

#### 

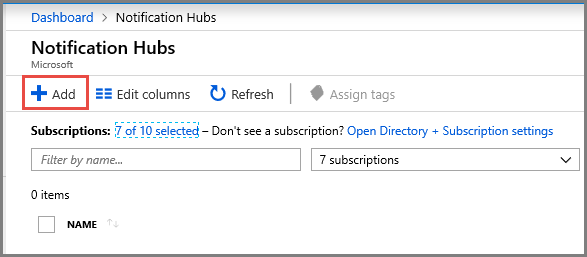
#### 

## **Configure a hub(create Azure notification service in azure portal)**

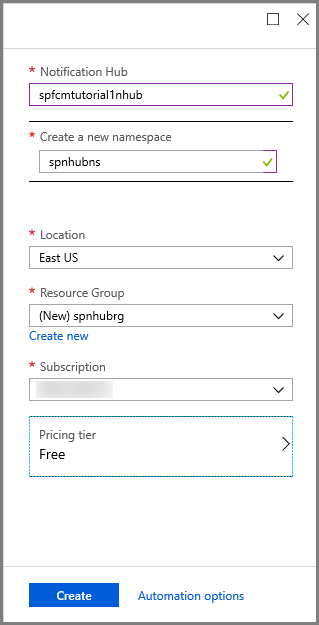
1. Sign in to the [Azure portal](https://portal.azure.com/).
2. Select **All services** on the left menu, and then select **Notification Hubs** in the **Mobile** section. Select the star icon next to the service name to add the service to the **FAVORITES** section on the left menu. After you add **Notification Hubs** to **FAVORITES**, select it on the left menu.



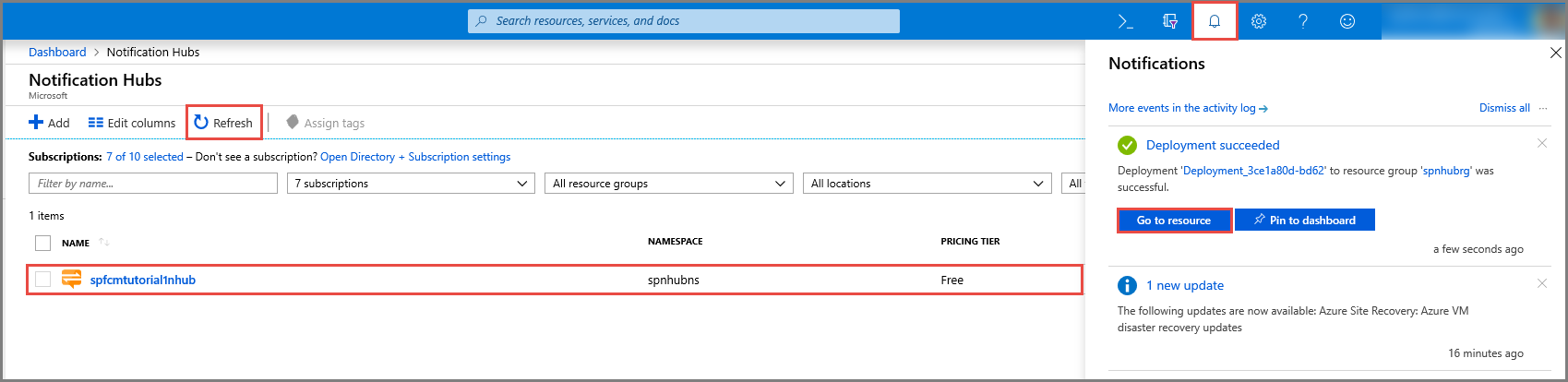
1. On the **Notification Hubs** page, select **Add** on the toolbar.



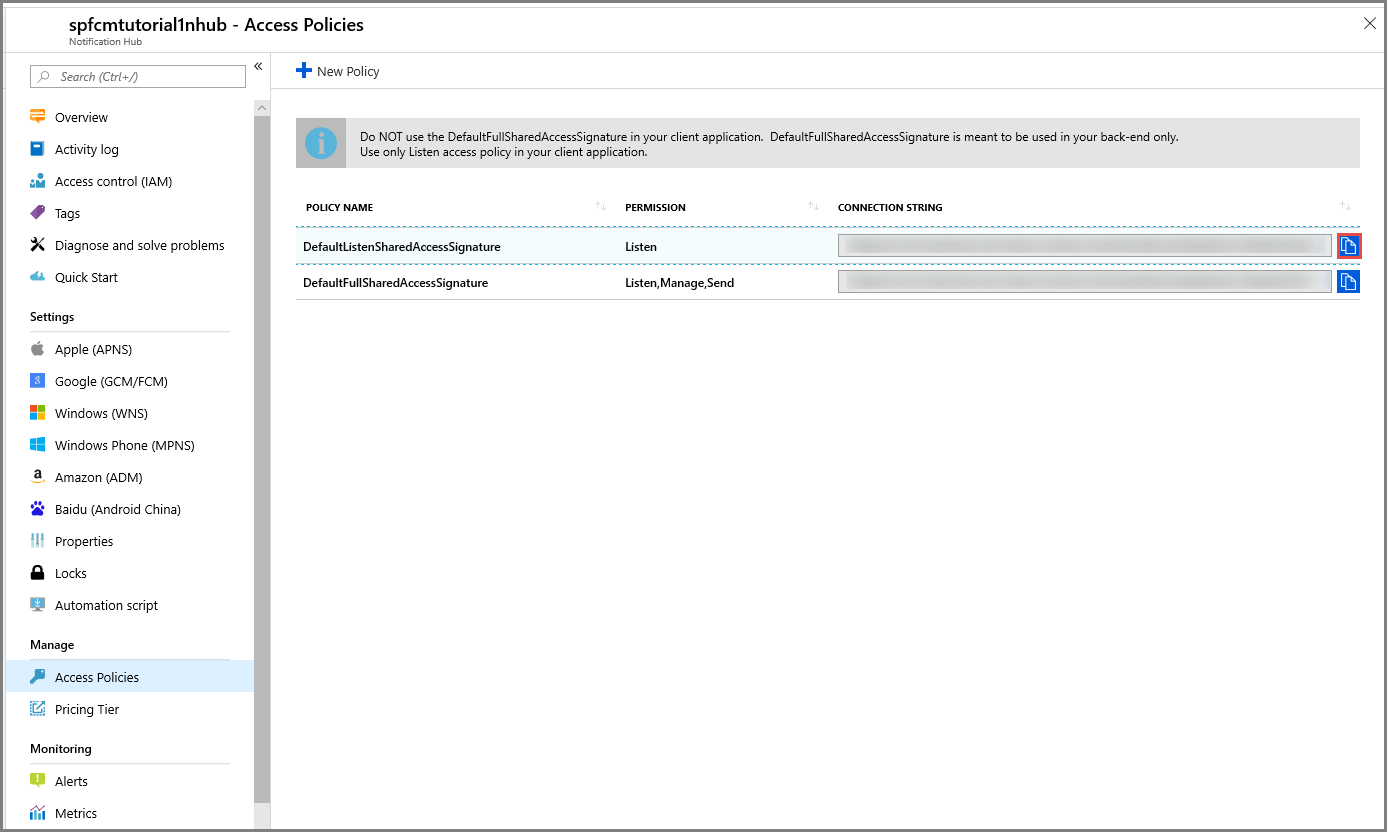
1. On the **Notification Hub** page, do the following steps:
   1. Enter a name in **Notification Hub**.
   2. Enter a name in **Create a new namespace**. A namespace contains one or more hubs.
   3. Select a value from the **Location** drop-down list box. This value specifies the location in which you want to create the hub.
2. Select an existing resource group in **Resource Group**, or create a name for a new resource group.
3. Select **Create**.



1. Select **Notifications** (the bell icon), and then select **Go to resource**. You can also refresh the list on the **Notification Hubs** page and select your hub.



1. Select **Access Policies** from the list. Note that the two connection strings are available to you. You'll need them later to handle push notifications.



### Configure Firebase Cloud Messaging settings for the hub

1. In the left pane, under **Settings,** select **Google (GCM/FCM)**.
2. Enter the **server key** for the FCM project that you saved earlier.
3. On the toolbar, select **Save**.

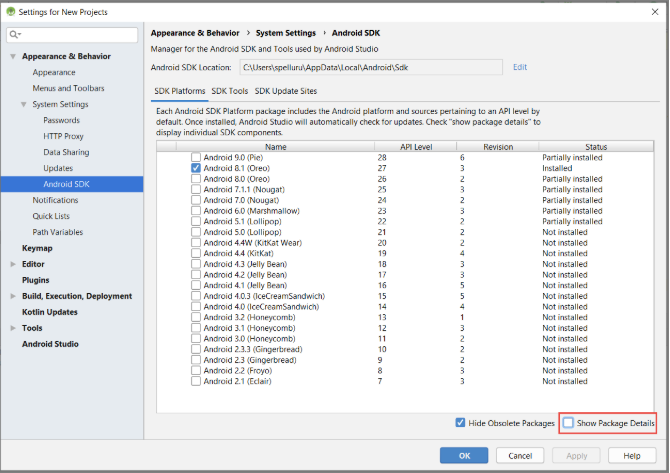


1. The Azure portal displays a message in alerts that the hub has been successfully updated. The **Save** button is disabled.

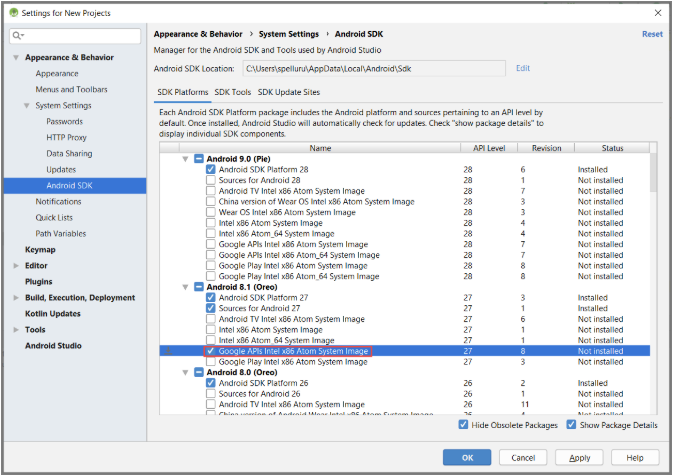
Your hub is now configured to work with Firebase Cloud Messaging. You also have the connection strings that are necessary to send notifications to a device and register an app to receive notifications.

## **Connect your app to the notification hub**

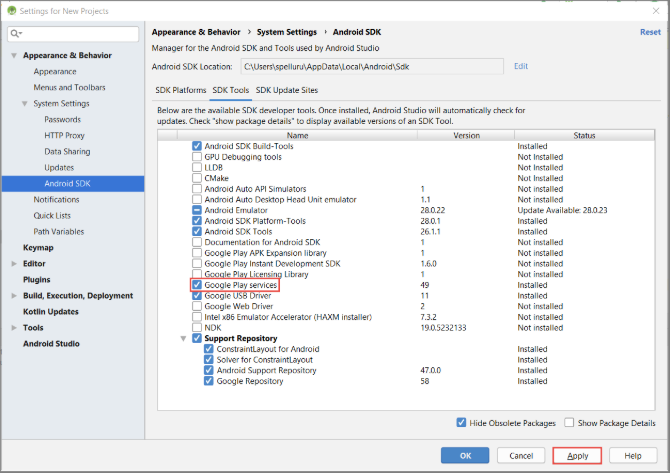
1. In Android Studio, select Tools on the menu, and then select SDK Manager.
2. Select the target version of the Android SDK that is used in your project. Then select Show Package Details.



1. Select **Google APIs**, if it's not already installed.



1. Switch to the **SDK Tools** tab. If you haven't already installed Google Play Services, select **Google Play Services** as shown in the following image. Then select **Apply** to install. Note the SDK path, for use in a later step.



1. If you see the Confirm Change dialog box, select OK. The Component Installer installs the requested components. Select Finish after the components are installed.
2. Select OK to close the Settings for New Projects dialog box.
3. Open the AndroidManifest.xml file, and then add the following tag to the application tag.

<meta-data android:name="com.google.android.gms.version"

android:value="@integer/google\_play\_services\_version" />

1. In the Build.Gradle file for the app, add the following lines in the dependencies section.

implementation 'com.microsoft.azure:notification-hubs-android-sdk:0.6@aar'

1. Add the following repository after the dependencies section.

repositories {

maven {

url "https://dl.bintray.com/microsoftazuremobile/SDK"

}

}

#### Add the following service definition to the AndroidManifest.xml file, inside the <application> tag.

#### <service

#### android:name=".RegistrationIntentService"

#### android:exported="false">

#### </service>

#### <service

#### android:name=".FirebaseService"

#### android:exported="false">

#### <intent-filter>

#### <action android:name="com.google.firebase.MESSAGING\_EVENT" />

#### </intent-filter>

#### </service>

#### Add the following necessary FCM-related permissions below the </application> tag.

#### <uses-permission android:name="android.permission.INTERNET"/>

#### <uses-permission android:name="android.permission.GET\_ACCOUNTS"/>

#### <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

#### In the Project View, expand app > src > main > java. Right-click your package folder under java, select New, and then select Java Class. Enter NotificationSettings for the name, and then select OK.

#### HubName : get the hub name from Azure portal

#### HubListenConnectionString: The DefaultListenAccessSignature connection string for your hub. You can copy that connection string by clicking Access Policies in your hub in the [Azure portal](https://portal.azure.com/).

public class NotificationSettings {

public static String HubName = "<Your HubName>";

public static String HubListenConnectionString = "<Enter your DefaultListenSharedAccessSignature connection string>";

}

#### Add another new class to your project named RegistrationIntentService. This class implements the IntentService interface. It also handles refreshing the FCM token and registering with the notification hub.

import android.app.IntentService;

import android.content.Intent;

import android.content.SharedPreferences;

import android.preference.PreferenceManager;

import android.util.Log;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.firebase.iid.FirebaseInstanceId;

import com.google.firebase.iid.InstanceIdResult;

import com.microsoft.windowsazure.messaging.NotificationHub;

import java.util.concurrent.TimeUnit;

public class RegistrationIntentService extends IntentService {

private static final String TAG = "RegIntentService";

String FCM\_token = null;

private NotificationHub hub;

public RegistrationIntentService() {

super(TAG);

}

@Override

protected void onHandleIntent(Intent intent) {

SharedPreferences sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this);

String resultString = null;

String regID = null;

String storedToken = null;

try {

FirebaseInstanceId.getInstance().getInstanceId().addOnSuccessListener(new OnSuccessListener<InstanceIdResult>() {

@Override

public void onSuccess(InstanceIdResult instanceIdResult) {

FCM\_token = instanceIdResult.getToken();

Log.d(TAG, "FCM Registration Token: " + FCM\_token);

}

});

TimeUnit.SECONDS.sleep(1);

// Storing the registration ID that indicates whether the generated token has been

// sent to your server. If it is not stored, send the token to your server.

// Otherwise, your server should have already received the token.

if (((regID=sharedPreferences.getString("registrationID", null)) == null)){

NotificationHub hub = new NotificationHub(NotificationSettings.HubName,

NotificationSettings.HubListenConnectionString, this);

Log.d(TAG, "Attempting a new registration with NH using FCM token : " + FCM\_token);

regID = hub.register(FCM\_token).getRegistrationId();

// If you want to use tags...

// Refer to : https://azure.microsoft.com/documentation/articles/notification-hubs-routing-tag-expressions/

// regID = hub.register(token, "tag1,tag2").getRegistrationId();

resultString = "New NH Registration Successfully - RegId : " + regID;

Log.d(TAG, resultString);

sharedPreferences.edit().putString("registrationID", regID ).apply();

sharedPreferences.edit().putString("FCMtoken", FCM\_token ).apply();

}

// Check to see if the token has been compromised and needs refreshing.

else if ((storedToken=sharedPreferences.getString("FCMtoken", "")) != FCM\_token) {

NotificationHub hub = new NotificationHub(NotificationSettings.HubName,

NotificationSettings.HubListenConnectionString, this);

Log.d(TAG, "NH Registration refreshing with token : " + FCM\_token);

regID = hub.register(FCM\_token).getRegistrationId();

// If you want to use tags...

// Refer to : https://azure.microsoft.com/documentation/articles/notification-hubs-routing-tag-expressions/

// regID = hub.register(token, "tag1,tag2").getRegistrationId();

resultString = "New NH Registration Successfully - RegId : " + regID;

Log.d(TAG, resultString);

sharedPreferences.edit().putString("registrationID", regID ).apply();

sharedPreferences.edit().putString("FCMtoken", FCM\_token ).apply();

}

else {

resultString = "Previously Registered Successfully - RegId : " + regID;

}

} catch (Exception e) {

Log.e(TAG, resultString="Failed to complete registration", e);

// If an exception happens while fetching the new token or updating registration data

// on a third-party server, this ensures that we'll attempt the update at a later time.

}

// Notify UI that registration has completed.

if (MainActivity.isVisible) {

MainActivity.mainActivity.ToastNotify(resultString);

}

}

}

#### In the MainActivity class, add the following import statements above the class declaration.

import com.google.android.gms.common.ConnectionResult;

import com.google.android.gms.common.GoogleApiAvailability;

import android.content.Intent;

import android.util.Log;

import android.widget.TextView;

#### import android.widget.Toast;

#### Add the following members at the top of the class. You use these fields to check the availability of Google Play Services as recommended by Google.

#### 

#### public static MainActivity mainActivity;

#### public static Boolean isVisible = false;

#### private static final String TAG = "MainActivity";

#### private static final int PLAY\_SERVICES\_RESOLUTION\_REQUEST = 9000;

#### In the MainActivity class, add the following method to check the availability of Google Play Services.

private boolean checkPlayServices() {

GoogleApiAvailability apiAvailability = GoogleApiAvailability.getInstance();

int resultCode = apiAvailability.isGooglePlayServicesAvailable(this);

if (resultCode != ConnectionResult.SUCCESS) {

if (apiAvailability.isUserResolvableError(resultCode)) {

apiAvailability.getErrorDialog(this, resultCode, PLAY\_SERVICES\_RESOLUTION\_REQUEST)

.show();

} else {

Log.i(TAG, "This device is not supported by Google Play Services.");

ToastNotify("This device is not supported by Google Play Services.");

finish();

}

return false;

}

return true;

}

#### In the MainActivity class, add the following code that checks for Google Play Services before calling the IntentService to get your FCM registration token and register with your hub:

public void registerWithNotificationHubs()

{

if (checkPlayServices()) {

// Start IntentService to register this application with FCM.

Intent intent = new Intent(this, RegistrationIntentService.class);

startService(intent);

}

}

#### In the OnCreate method of the MainActivity class, add the following code to start the registration process when the activity is created:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mainActivity = this;

registerWithNotificationHubs();

FirebaseService.createChannelAndHandleNotifications(getApplicationContext());

}

#### To verify app state and report status in your app, add these additional methods to MainActivity:

#### 

@Override

protected void onStart() {

super.onStart();

isVisible = true;

}

@Override

protected void onPause() {

super.onPause();

isVisible = false;

}

@Override

protected void onResume() {

super.onResume();

isVisible = true;

}

@Override

protected void onStop() {

super.onStop();

isVisible = false;

}

public void ToastNotify(final String notificationMessage) {

runOnUiThread(new Runnable() {

@Override

public void run() {

Toast.makeText(MainActivity.this, notificationMessage, Toast.LENGTH\_LONG).show();

TextView helloText = (TextView) findViewById(R.id.text\_hello);

helloText.setText(notificationMessage);

}

});

}

1. The ToastNotify method uses the "Hello World" TextView control to report status and notifications persistently in the app. In your **res** > **layout** > **activity\_main.xml** layout, add the following ID for that control.

#### android:id="@+id/text\_hello"

#### 

#### Next you add a subclass for the receiver that you defined in AndroidManifest.xml. Add another new class to your project named FirebaseService.

#### This code overrides the onMessageReceived method and reports notifications that are received. it also sends the push notification to the Android notification manager by using the sendNotification() method. Call the sendNotification() method when the app isn't running and a notification is received.

import com.google.firebase.messaging.FirebaseMessagingService;

import com.google.firebase.messaging.RemoteMessage;

import android.util.Log;

import android.app.NotificationChannel;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.content.Context;

import android.content.Intent;

import android.media.RingtoneManager;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import androidx.core.app.NotificationCompat;

public class FirebaseService extends FirebaseMessagingService

{

private String TAG = "FirebaseService";

public static final String NOTIFICATION\_CHANNEL\_ID = "nh-demo-channel-id";

public static final String NOTIFICATION\_CHANNEL\_NAME = "Notification Hubs Demo Channel";

public static final String NOTIFICATION\_CHANNEL\_DESCRIPTION = "Notification Hubs Demo Channel";

public static final int NOTIFICATION\_ID = 1;

private NotificationManager mNotificationManager;

NotificationCompat.Builder builder;

static Context ctx;

@Override

public void onMessageReceived(RemoteMessage remoteMessage) {

// ...

// TODO(developer): Handle FCM messages here.

// Not getting messages here? See why this may be: https://goo.gl/39bRNJ

Log.d(TAG, "From: " + remoteMessage.getFrom());

String nhMessage;

// Check if message contains a notification payload.

if (remoteMessage.getNotification() != null) {

Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());

nhMessage = remoteMessage.getNotification().getBody();

}

else {

nhMessage = remoteMessage.getData().values().iterator().next();

}

// Also if you intend on generating your own notifications as a result of a received FCM

// message, here is where that should be initiated. See sendNotification method below.

if (MainActivity.isVisible) {

MainActivity.mainActivity.ToastNotify(nhMessage);

}

sendNotification(nhMessage);

}

private void sendNotification(String msg) {

Intent intent = new Intent(ctx, MainActivity.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);

mNotificationManager = (NotificationManager)

ctx.getSystemService(Context.NOTIFICATION\_SERVICE);

PendingIntent contentIntent = PendingIntent.getActivity(ctx, 0,

intent, PendingIntent.FLAG\_ONE\_SHOT);

Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE\_NOTIFICATION);

NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(

ctx,

NOTIFICATION\_CHANNEL\_ID)

.setContentText(msg)

.setPriority(NotificationCompat.PRIORITY\_HIGH)

.setSmallIcon(android.R.drawable.ic\_popup\_reminder)

.setBadgeIconType(NotificationCompat.BADGE\_ICON\_SMALL);

notificationBuilder.setContentIntent(contentIntent);

mNotificationManager.notify(NOTIFICATION\_ID, notificationBuilder.build());

}

public static void createChannelAndHandleNotifications(Context context) {

ctx = context;

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {

NotificationChannel channel = new NotificationChannel(

NOTIFICATION\_CHANNEL\_ID,

NOTIFICATION\_CHANNEL\_NAME,

NotificationManager.IMPORTANCE\_HIGH);

channel.setDescription(NOTIFICATION\_CHANNEL\_DESCRIPTION);

channel.setShowBadge(true);

NotificationManager notificationManager = context.getSystemService(NotificationManager.class);

notificationManager.createNotificationChannel(channel);

}

}

}

#### Ensure you have a virtual device(Emulator) for running the app and Run the app on your selected device and verify that it registers successfully with the hub.

#### When you open installed app on emulator it will show a registration id of our app with FCM will be displayed.

#### 

#### Open the res/layout/activity\_main.xml file, and replace the content with the following:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.breakingnews.MainActivity"

android:orientation="vertical">

<CheckBox

android:id="@+id/worldBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/label\_world" />

<CheckBox

android:id="@+id/politicsBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/label\_politics" />

<CheckBox

android:id="@+id/businessBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/label\_business" />

<CheckBox

android:id="@+id/technologyBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/label\_technology" />

<CheckBox

android:id="@+id/scienceBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/label\_science" />

<CheckBox

android:id="@+id/sportsBox"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/label\_sports" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="subscribe"

android:text="@string/button\_subscribe" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

android:id="@+id/text\_hello"

/>

</LinearLayout>

#### Open the res/values/strings.xml file and add the following lines:

<string name="button\_subscribe">Subscribe</string>

<string name="label\_world">World</string>

<string name="label\_politics">Politics</string>

<string name="label\_business">Business</string>

<string name="label\_technology">Technology</string>

<string name="label\_science">Science</string>

<string name="label\_sports">Sports</string>

#### Create a class Notifications in the same package as your MainActivity class.

import java.util.HashSet;

import java.util.Set;

import java.util.concurrent.TimeUnit;

import android.content.Context;

import android.content.SharedPreferences;

import android.os.AsyncTask;

import android.util.Log;

import android.widget.Toast;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.firebase.iid.FirebaseInstanceId;

import com.google.firebase.iid.InstanceIdResult;

import com.microsoft.windowsazure.messaging.NotificationHub;

public class Notifications {

private static final String PREFS\_NAME = "BreakingNewsCategories";

private FirebaseInstanceId fcm;

private NotificationHub hub;

private Context context;

private String senderId;

public static String FCM\_token = "";

private static final String TAG = "Notifications";

public Notifications(Context context, String hubName, String listenConnectionString) {

this.context = context;

this.senderId = senderId;

fcm = FirebaseInstanceId.getInstance();

hub = new NotificationHub(hubName, listenConnectionString, context);

}

public void storeCategoriesAndSubscribe(Set<String> categories)

{

SharedPreferences settings = context.getSharedPreferences(PREFS\_NAME, 0);

settings.edit().putStringSet("categories", categories).commit();

subscribeToCategories(categories);

}

public Set<String> retrieveCategories() {

SharedPreferences settings = context.getSharedPreferences(PREFS\_NAME, 0);

return settings.getStringSet("categories", new HashSet<String>());

}

public void subscribeToCategories(final Set<String> categories) {

new AsyncTask<Object, Object, Object>() {

@Override

protected Object doInBackground(Object... params) {

try {

FirebaseInstanceId.getInstance().getInstanceId().addOnSuccessListener(new OnSuccessListener<InstanceIdResult>() {

@Override

public void onSuccess(InstanceIdResult instanceIdResult) {

FCM\_token = instanceIdResult.getToken();

Log.d(TAG, "FCM Registration Token: " + FCM\_token);

}

});

TimeUnit.SECONDS.sleep(1);

String templateBodyFCM = "{\"data\":{\"message\":\"$(messageParam)\"}}";

hub.registerTemplate(FCM\_token,"simpleFCMTemplate", templateBodyFCM,

categories.toArray(new String[categories.size()]));

} catch (Exception e) {

Log.e("MainActivity", "Failed to register - " + e.getMessage());

return e;

}

return null;

}

protected void onPostExecute(Object result) {

String message = "Subscribed for categories: "

+ categories.toString();

Toast.makeText(context, message,

Toast.LENGTH\_LONG).show();

}

}.execute(null, null, null);

}

}

#### In your MainActivity class, add a field for Notifications

#### private Notifications notifications;

#### Then, update the onCreate method as shown in the following code. You register with Notification Hubs in the **subscribeToCategories** method of the **Notifications** class.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mainActivity = this;

FirebaseService.createChannelAndHandleNotifications(getApplicationContext());

notifications = new Notifications(this, NotificationSettings.HubName, NotificationSettings.HubListenConnectionString);

notifications.subscribeToCategories(notifications.retrieveCategories());

}

#### Then, add the following imports

import android.widget.CheckBox;

import java.util.HashSet;

import java.util.Set;

#### import android.view.View;

#### Add the following subscribe method to handle the subscribe button click event:

public void subscribe(View sender) {

final Set<String> categories = new HashSet<String>();

CheckBox world = (CheckBox) findViewById(R.id.worldBox);

if (world.isChecked())

categories.add("world");

CheckBox politics = (CheckBox) findViewById(R.id.politicsBox);

if (politics.isChecked())

categories.add("politics");

CheckBox business = (CheckBox) findViewById(R.id.businessBox);

if (business.isChecked())

categories.add("business");

CheckBox technology = (CheckBox) findViewById(R.id.technologyBox);

if (technology.isChecked())

categories.add("technology");

CheckBox science = (CheckBox) findViewById(R.id.scienceBox);

if (science.isChecked())

categories.add("science");

CheckBox sports = (CheckBox) findViewById(R.id.sportsBox);

if (sports.isChecked())

categories.add("sports");

notifications.storeCategoriesAndSubscribe(categories);

}

#### Confirm that the following code is at the end of the onCreate method in the MainActivity class:

#### notifications.subscribeToCategories(notifications.retrieveCategories());

#### Then update the onStart() method of the MainActivity class as follows:

#### @Override

#### protected void onStart() {

#### super.onStart();

#### isVisible = true;

#### Set<String> categories = notifications.retrieveCategories();

#### CheckBox world = (CheckBox) findViewById(R.id.worldBox);

#### world.setChecked(categories.contains("world"));

#### CheckBox politics = (CheckBox) findViewById(R.id.politicsBox);

#### politics.setChecked(categories.contains("politics"));

#### CheckBox business = (CheckBox) findViewById(R.id.businessBox);

#### business.setChecked(categories.contains("business"));

#### CheckBox technology = (CheckBox) findViewById(R.id.technologyBox);

#### technology.setChecked(categories.contains("technology"));

#### CheckBox science = (CheckBox) findViewById(R.id.scienceBox);

#### science.setChecked(categories.contains("science"));

#### CheckBox sports = (CheckBox) findViewById(R.id.sportsBox);

#### sports.setChecked(categories.contains("sports"));

#### }

#### 

## **Send tagged notifications from Backend Console application**

1. In Visual Studio, create a new Visual C# console application:
   1. On the menu, select **File** > **New** > **Project**.
   2. In **Create a new project**, select **Console App (.NET Framework)** for C# in the list of templates, and select **Next**.
   3. Enter a name for the app.
   4. For **Solution**, choose **Add to solution**, and select **Create** to create the project.
2. Select **Tools** > **NuGet Package Manager** > **Package Manager Console** and then, in the console window, run the following command:

Install-Package Microsoft.Azure.NotificationHubs

1. Open the *Program.cs* file, and add the following using statement:

using Microsoft.Azure.NotificationHubs;

1. In the Program class, add the following method

private static async void SendTemplateNotificationAsync()

{

// Define the notification hub.

NotificationHubClient hub = NotificationHubClient.CreateClientFromConnectionString("<connection string with full access>", "<hub name>");

// Apple requires the apns-push-type header for all requests

var headers = new Dictionary<string, string> {{"apns-push-type", "alert"}};

// Create an array of breaking news categories.

var categories = new string[] { "World", "Politics", "Business", "Technology", "Science", "Sports"};

// Send the notification as a template notification. All template registrations that contain

// "messageParam" and the proper tags will receive the notifications.

// This includes APNS, GCM/FCM, WNS, and MPNS template registrations.

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

foreach (var category in categories)

{

templateParams["messageParam"] = "Breaking " + category + " News!";

await hub.SendTemplateNotificationAsync(templateParams, category);

}

}

#### In the preceding code, replace the <hub name> and <connection string with full access> placeholders with your notification hub name and the connection string for DefaultFullSharedAccessSignature from the dashboard of your notification hub.

#### In the Main() method, add the following lines:

#### SendTemplateNotificationAsync();

#### Console.ReadLine();

#### Build the console app.

## **Test send notification from the console Application**

1. In  Android Studio, run the app on your Android device or emulator. The app UI provides a set of toggles that lets you choose the categories to subscribe to.
2. Enable one or more categories toggles, then click **Subscribe**. The app converts the selected categories into tags and requests a new device registration for the selected tags from the notification hub. The registered categories are returned and displayed in a toast notification.

#### 

#### Run the .NET console app, which sends notifications for each category. Notifications for the selected categories appear as toast notifications.

#### 