Documentation for Notifications

Notifications Team Omega

April 2020

1 Introduction

This document contains an overview of the functions we as the notifications team have implemented and used to create the local notifications and the global push notifications for the app. The email sending functionality and usage are defined and explained in a separate document, as this will be contained on the sever instead of on the app itself.

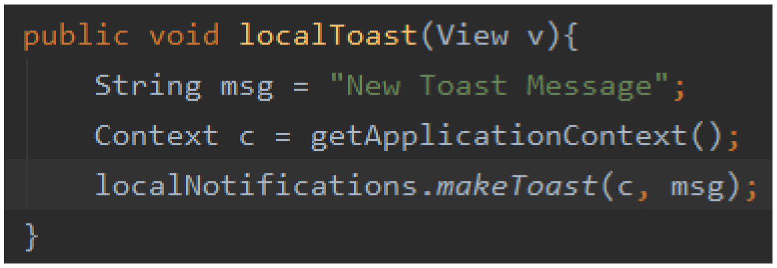
2 Local Notifications

For the local notifications we used Toasts that will notify the users of any actions that have been completed or that have been started, these will pop-up when they are triggered by the app with a message that matches the action that it was triggered by. A class has been created that will allow other teams to call the function with the message that they desire to be show and they call the function and the toast will notify the user.

2.1 localNotifications.java

The **localNotifications** class contains a function called *makeToast(Context c, String msg)* and as show it takes two parameters the first being the **Context** of the application, and the second being a **String** that will be the message that will be displayed as the notification.

In order to make a local notification, the team using our **localNotifications** class creates a new **Context** variable by calling the ***getApplicationContext()*** and then creating a message to be sent, then calling the ***makeToast(Context c, String msg)*** that will display the toast with the created message as a Toast.



```
public void localToast(View v){
    String msg = "New Toast Message";
    Context c = getApplicationContext();
    localNotifications.makeToast(c, msg);
}
```

Figure 1: Calling ***makeToast(Context c, String msg)*** to display Toast.

3 Global Push Notifications

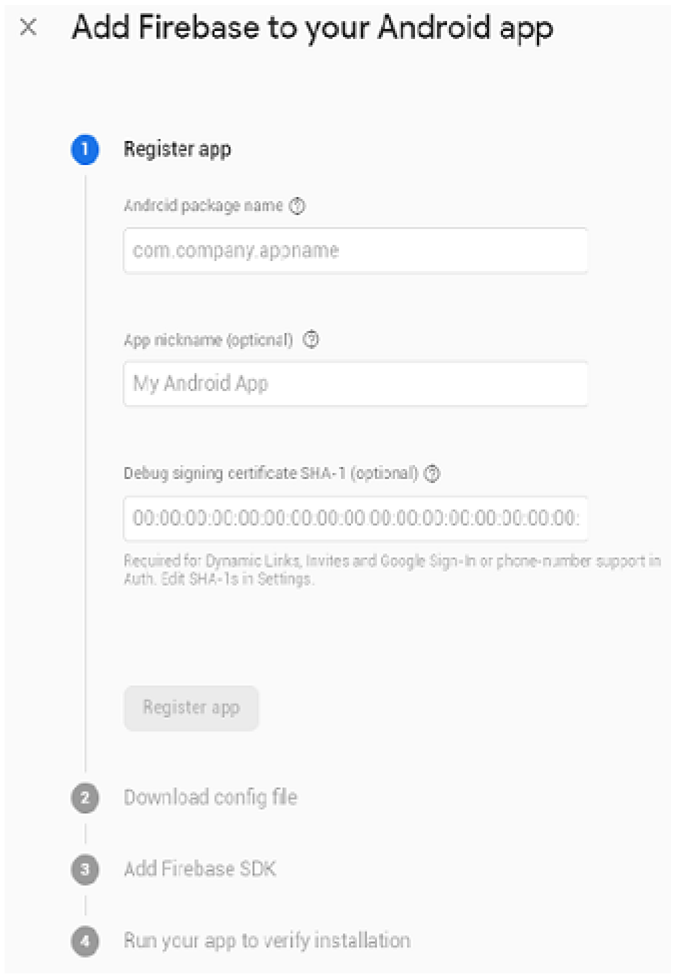
In order to receive push notifications on our android app we have implemented Firebase which sends the messages we create directly to the application. Firebase also allows for us to use certain analytic tools to see how effective the notifications that are being sent are.

3.1 Firebase

Firstly we need to create a new project in Firebase for our application and complete the creation. Once done we open the project to allow us to add an android application to the project. In the project console we click on the android icon to add an application to the project.

We do this by adding the package name of our application as well as giving our app a nickname as shown in Figure 2. After that we download the google-services.json file and add it to the app folder of our Android Studio project folder. We then open the Android Studio project as we nee to edit the Gradle and Manifest file to allow Firebase to actually work. And allow the application to receive the Notifications.

After we have added the file in the app directory we open the Project build.gradle and modify it as shown in Figure 4, and we modify the App level Gradle as shown in Figure 5. After we have modified the Gradle files we press the ***Sync Now*** prompt that appears. After that rebuild the app and run it on an AVD, then check that the app has communicated with our the firbase server.



Add Firebase to your Android app

1 Register app

Android package name

com.company.appname

App nickname (optional)

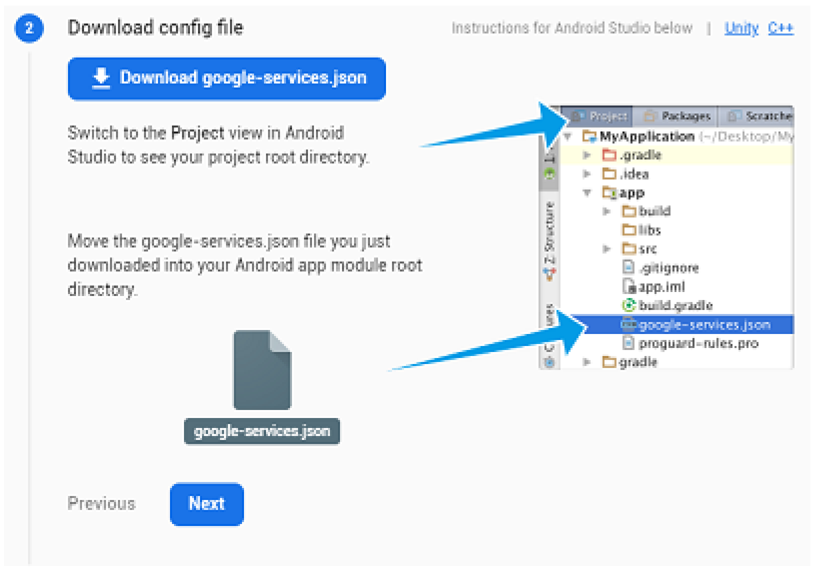
My Android App

Debug signing certificate SHA-1 (optional)

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:

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

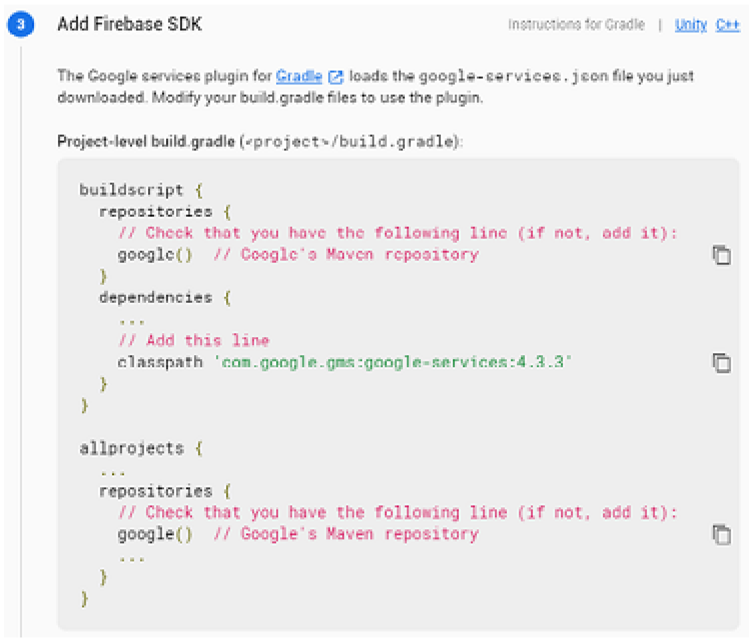
Figure 2: Add app to Firebase



Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

Figure 3: Download and add google-services.json file.

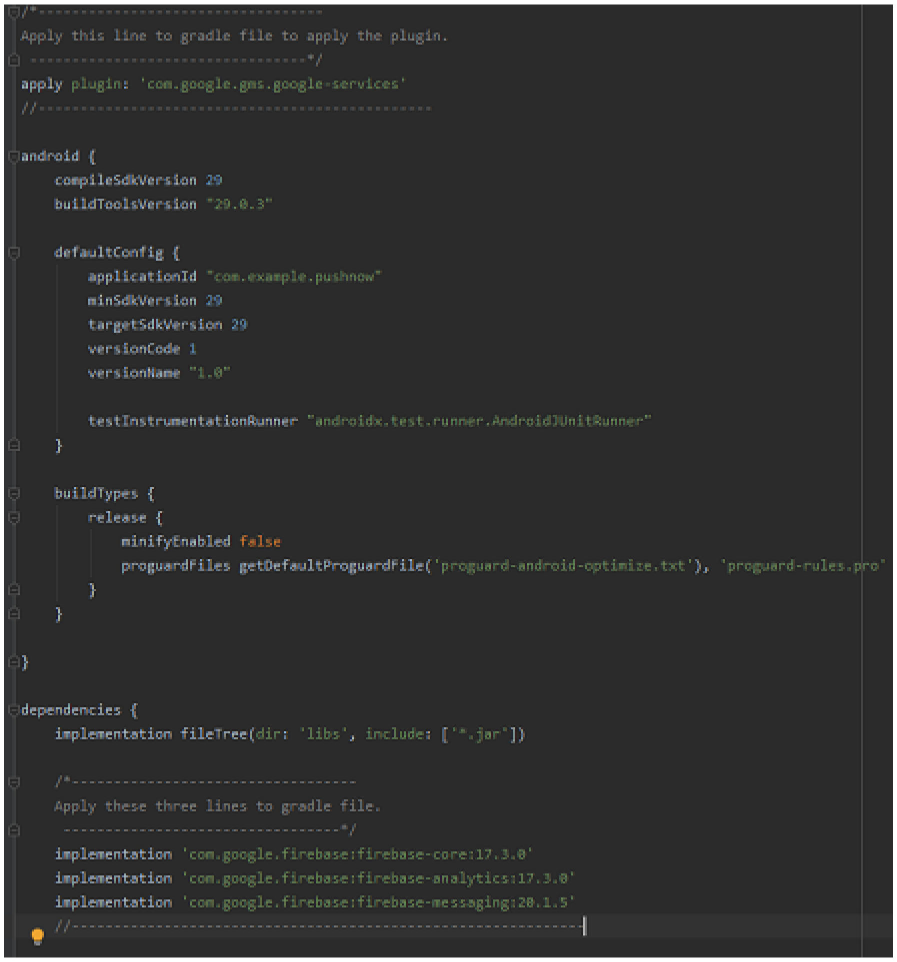


The Google services plugin for Gradle loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.3'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

Figure 4: Download and add google-services.json file.

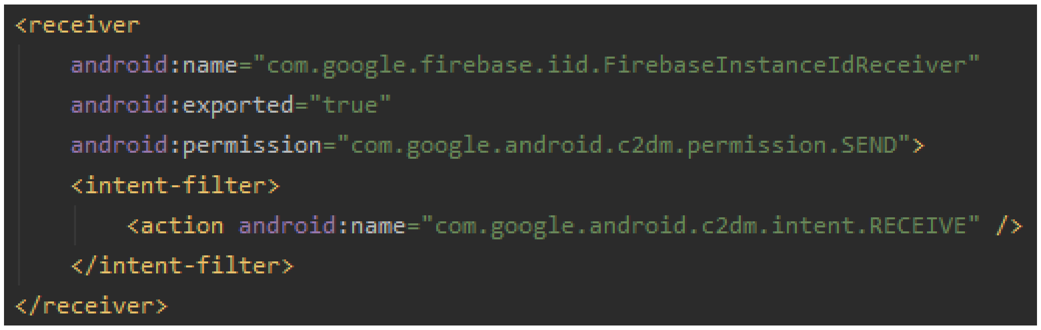


```
/*---------------------------------
Apply this line to gradle file to apply the plugin.
 --------------------------------*/
apply plugin: 'com.google.gms.google-services'
//--------------------------------------------

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "com.example.pushnow"
        minSdkVersion 29
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    /*---------------------------------
    Apply these three lines to gradle file.
     --------------------------------*/
    implementation 'com.google.firebase:firebase-core:17.3.0'
    implementation 'com.google.firebase:firebase-analytics:17.3.0'
    implementation 'com.google.firebase:firebase-messaging:20.1.5'
    //--------------------------------------------------------------
```

Figure 5: Download and add google-services.json file.

3.2 Receiving Messages with Android

The final part of the receiving messages on the device with the application installed is to update the AndroidManifest.xml file to allow the application to receive messages.



```xml
<receiver
    android:name="com.google.firebase.iid.FirebaseInstanceIdReceiver"
    android:exported="true"
    android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
    </intent-filter>
</receiver>
```

We also need to ensure that we have added the internet permission in the manifest file.