53. Integrating Firebase Support into an Android Studio Project

The next chapter (*An Android 7 Firebase Remote Notification Tutorial*) will use Firebase to send remote notifications to an Android app. Before Firebase can be used, however, there are a number of steps that must be taken within both Firebase and the Android Studio project. This chapter is intended to serve as a brief introduction to Firebase and to outline the integration of Firebase notifications into an Android Studio project.

53.1 What is Firebase?

Before being acquired by Google in 2014, Firebase started out as an independent company providing cloud-based solutions such as real-time database, analytics, messaging, notification and crash reporting services to web and mobile app developers. Firebase essentially consists of a set of cloud services, programming interfaces and libraries combined with a web-based console through which the services are managed.

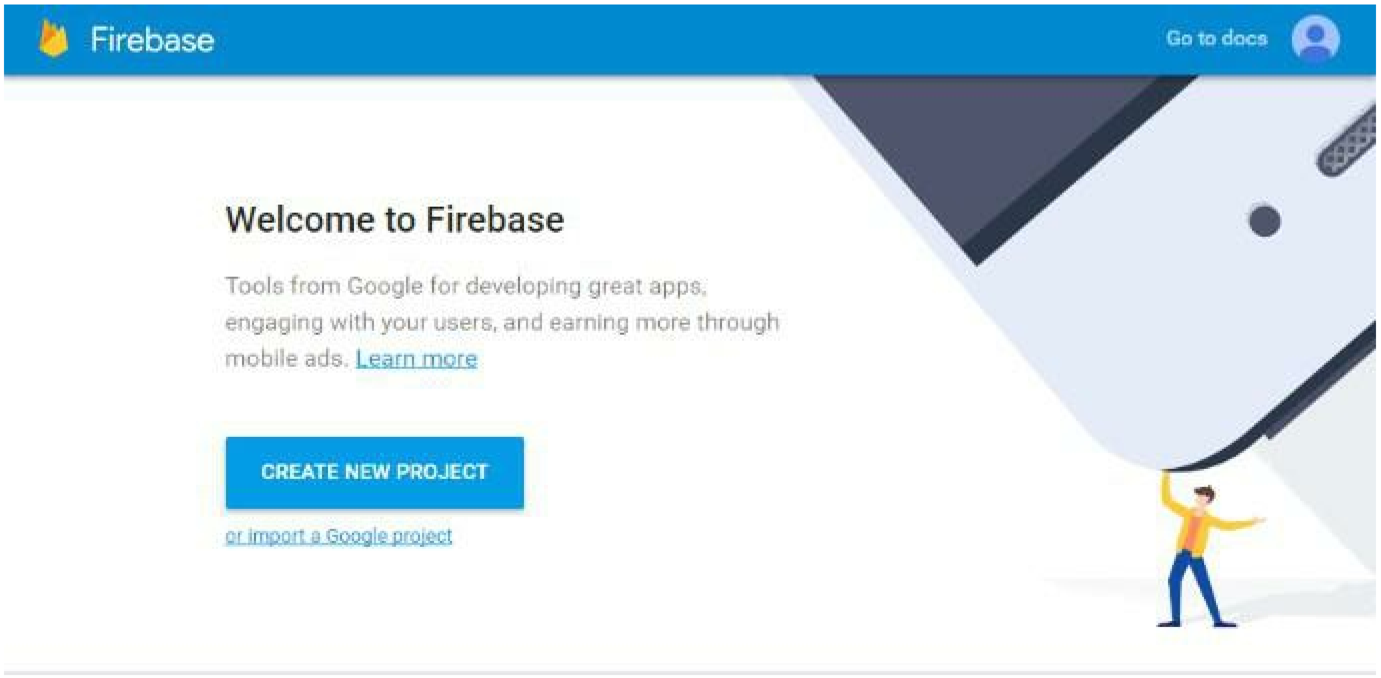
One of the services provided by Firebase, and the topic of the next chapter, is the ability to send remote notifications to Android apps. Before trying out Firebase remote notifications, however, some initial setup steps need to be taken.

53.2 Signing in to Firebase

All that is required to use Firebase is a Google account. To begin using Firebase, start by navigating to the following URL:

https://firebase.google.com/

Once the page has loaded, click on the *Get Started for Free* button to access your Firebase console as illustrated in Figure 53-1 below:



Welcome to Firebase

Tools from Google for developing great apps, engaging with your users, and earning more through mobile ads. Learn more

Figure 53-1

Firebase projects can be created and connected to an Android Studio project either using the Firebase console or via the Android Studio Firebase plugin. By far the most convenient option, and the approach taken in this chapter, is to use the plugin. First, however, an Android Studio project needs to be created.

53.3 Creating the FirebaseNotify Project

Start Android Studio and create a new project, entering *FirebaseNotify* into the Application name field and *ebookfrenzy.com* as the Company Domain setting before clicking on the *Next* button.

On the form factors screen, enable the *Phone and Tablet* option and set the minimum SDK setting to API 24: Android 7.0 (Nougat). Continue through the setup screens, requesting the creation of an Empty Activity named *FirebaseNotifyActivity* with a corresponding layout file named *activity_firebase_notify*.

53.4 Configuring the User Interface

For this example, the only change required to the layout resource file is to assign an ID to the default "Hello World!" TextView object. Load the *firebase_notify_activity.xml* file into the Layout Editor tool, select the TextView widget and, using the Properties panel, set the ID to *myTextView*.

53.5 Connecting the Project to Firebase

An Android Studio project can be configured to support Firebase using the Firebase plugin. This is accessed from within Android Studio by selecting the *Tools -> Firebase* menu option. This will display the Assistant tool window in the right-hand panel of the Android Studio main window. Within this panel, select the *Notifications* option as illustrated in Figure 53-2:

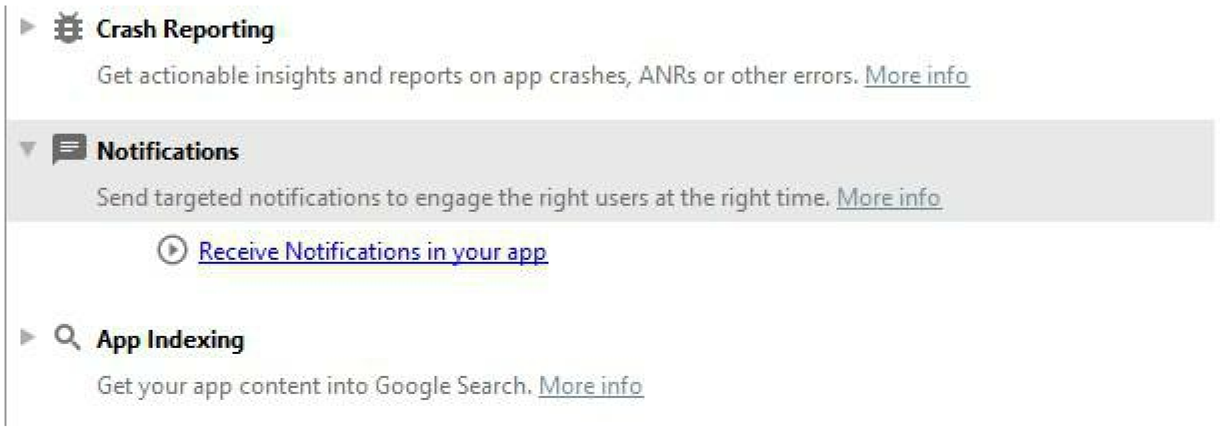
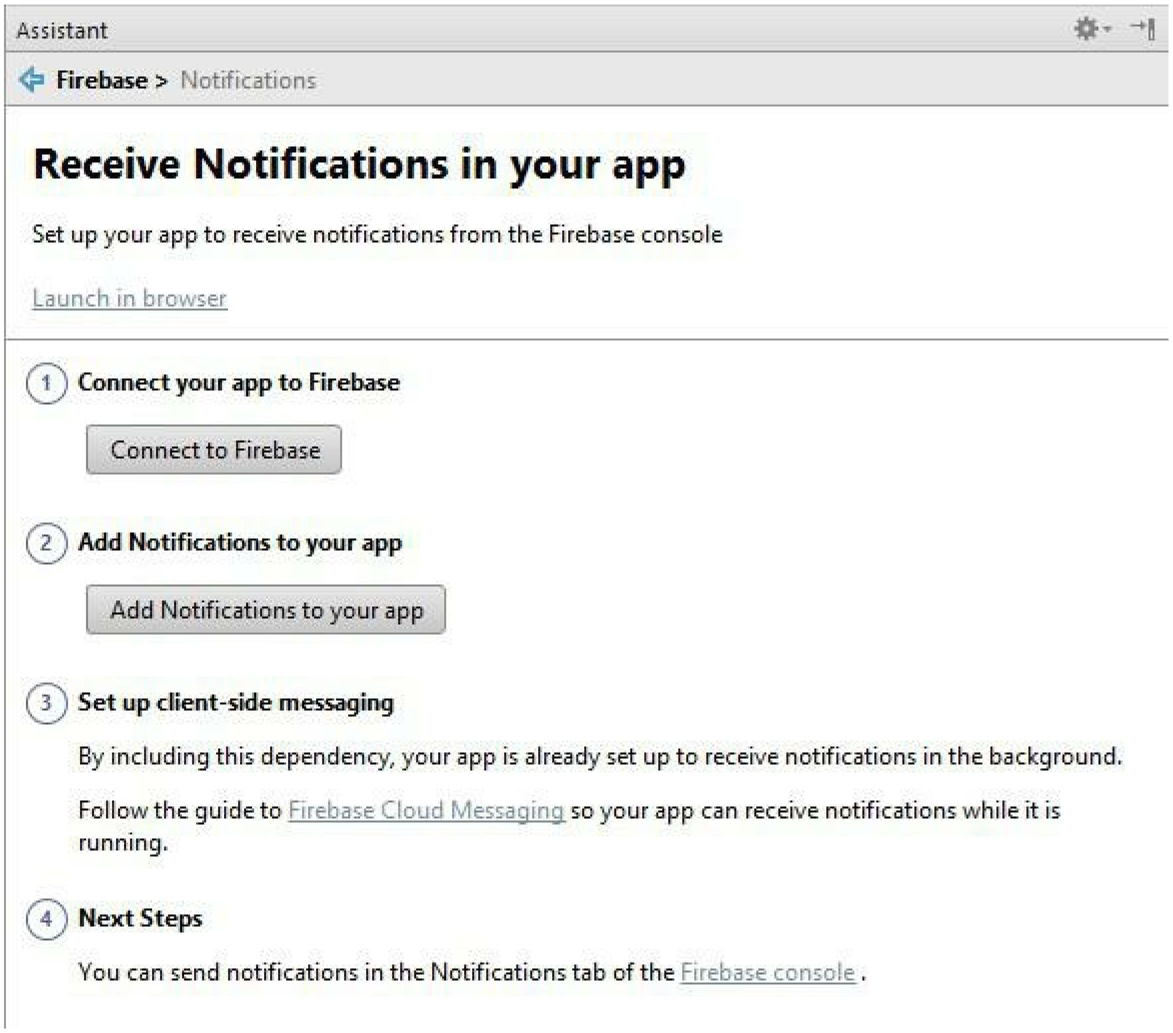

Crash Reporting
Get actionable insights and reports on app crashes, ANRs or other errors. More info
Notifications
Send targeted notifications to engage the right users at the right time. More info
Receive Notifications in your app
App Indexing
Get your app content into Google Search. More info

Figure 53-2

Within the Notifications section of the panel, click on the *Receive notifications in your app* link. This will display the Firebase Notifications panel shown in Figure 53-3 below:


Assistant

Firebase > Notifications

Receive Notifications in your app

Set up your app to receive notifications from the Firebase console

Launch in browser

1 Connect your app to Firebase

Connect to Firebase

2 Add Notifications to your app

Add Notifications to your app

3 Set up client-side messaging

By including this dependency, your app is already set up to receive notifications in the background.

Follow the guide to Firebase Cloud Messaging so your app can receive notifications while it is running.

4 Next Steps

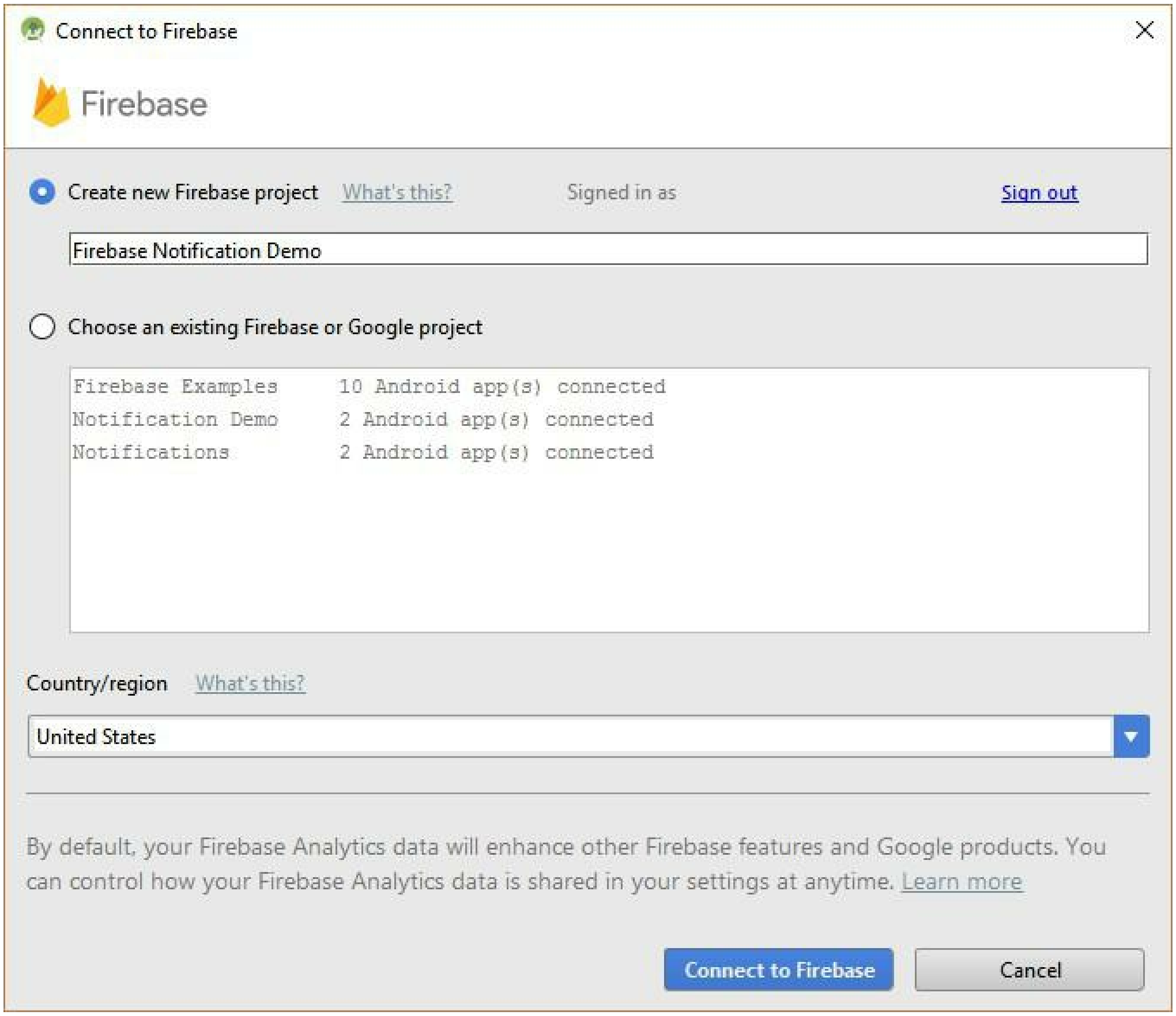
You can send notifications in the Notifications tab of the Firebase console .

Figure 53-3

In order to create a new Firebase project and associate it with the current Android Studio project, the app must be connected to Firebase. To achieve this, click on the *Connect to Firebase* button. If this is the first time you have established a connection to Firebase from within Android Studio, a browser window will appear requesting permission to access your Google account which you must accept to continue.

53.6 **Creating a New Firebase Project**

The app project in Android Studio must be connected with a Firebase project to be able to make use of services such as remote notifications. Once the Android Studio project has established communication with Firebase, a dialog (Figure 53-4) will appear providing the option to create a new Firebase project, or connect to an existing one.


Connect to Firebase

Firebase

(● Create new Firebase project) What's this? Signed in as Sign out

Firebase Notification Demo

○ Choose an existing Firebase or Google project

```
Firebase Examples     10 Android app(s) connected
Notification Demo     2 Android app(s) connected
Notifications         2 Android app(s) connected
```

Country/region What's this?

United States

By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. Learn more

Figure 53-4

Keep the *Create new Firebase Project* option selected and enter *Firebase Notification Demo* into the text field. Select your country from the drop-down menu, click on the *Connect to Firebase* button and wait for notification from Android Studio that the connection has completed successfully.

53.7 **The google-services.json File**

As part of the process of connecting the app to the Firebase project, a file named *google-services.json* will have been added to the Android Studio project. To locate this file, switch the Project tool window from Android to Project mode using the drop-down menu:

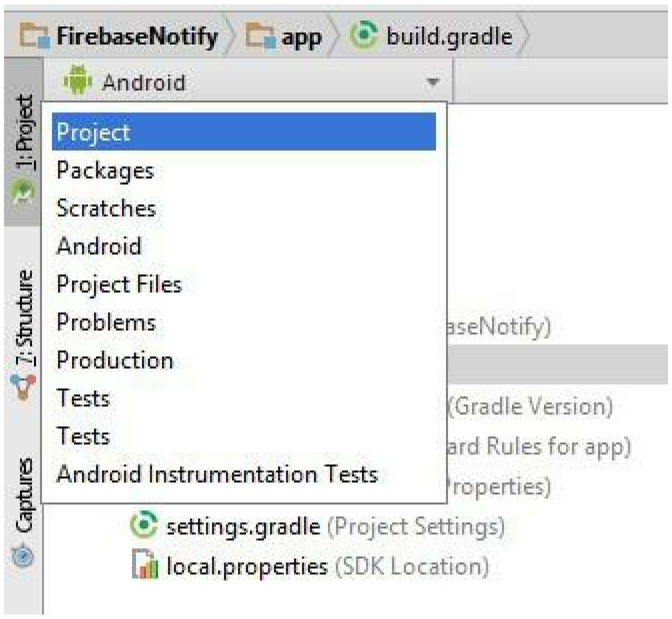


Figure 53-5

Once the Project tool window is in Project mode, unfold the project levels until the *app* folder comes into view and look for the *google-services.json* file in the project hierarchy as highlighted in Figure 53-6:

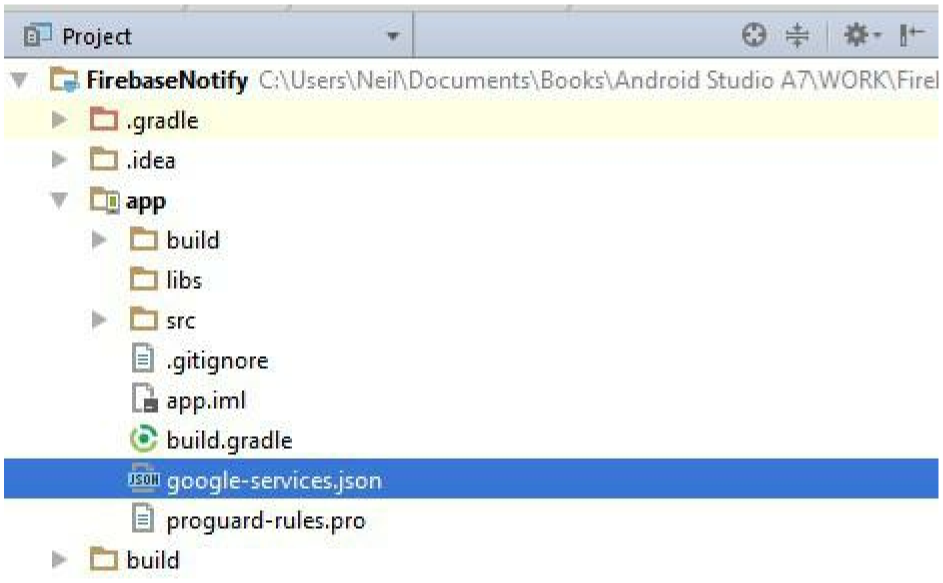
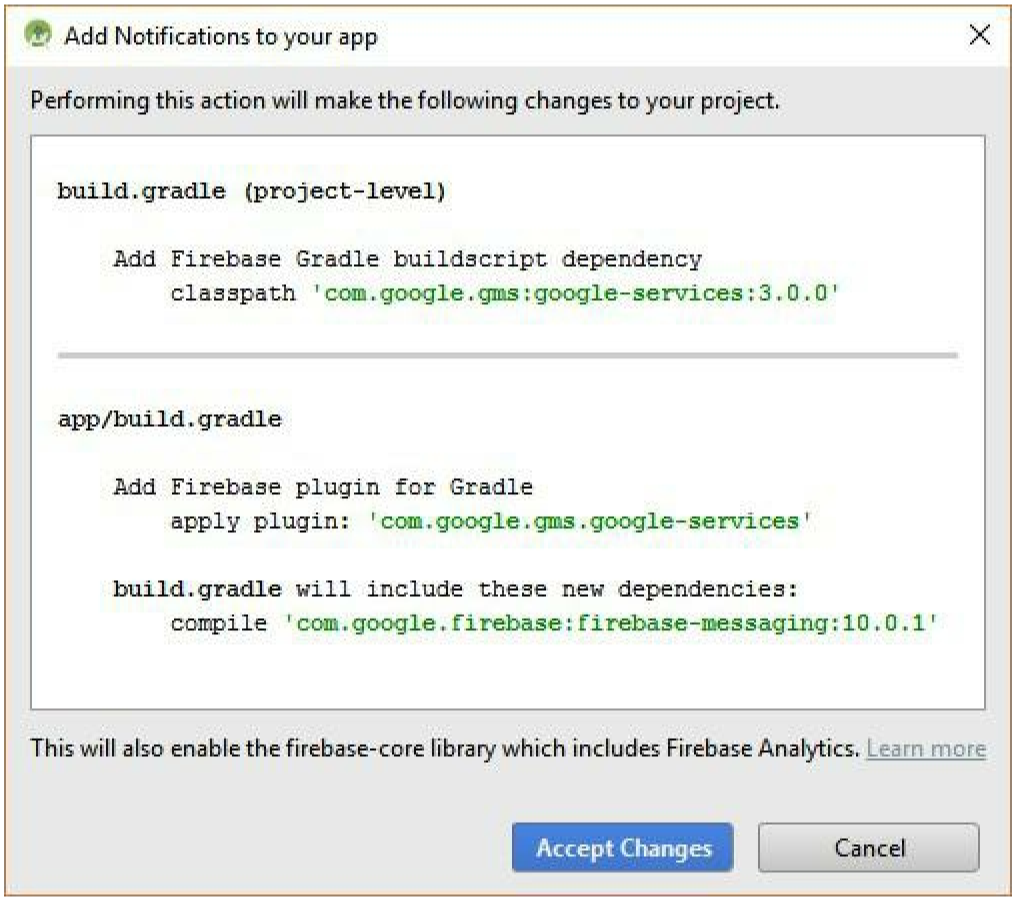


Figure 53-6

This configuration file contains the information that uniquely identifies your app within the Firebase ecosystem and its presence within the project is essential for Firebase services to work. Having confirmed the presence of this file, revert the Project tool window to Android mode before continuing.

53.8 Adding the Firebase Libraries

A number of libraries need to be added to the project in order to fully support Firebase notifications. These dependences may be added to the project automatically by clicking on the *Add Notifications to your app* button located in the Firebase Assistant panel as outlined in Figure 53-3 above. Clicking this button will display the following dialog:


Add Notifications to your app

Performing this action will make the following changes to your project.

```
build.gradle (project-level)

    Add Firebase Gradle buildscript dependency
        classpath 'com.google.gms:google-services:3.0.0'

app/build.gradle

    Add Firebase plugin for Gradle
        apply plugin: 'com.google.gms.google-services'

    build.gradle will include these new dependencies:
        compile 'com.google.firebase:firebase-messaging:10.0.1'
```

This will also enable the firebase-core library which includes Firebase Analytics. Learn more

Figure 53-7

Click on the *Accept Changes* button to make the changes to the project. Android Studio will add the library dependences and synchronize the project build files to reflect the changes.

The first change made by the assistant during this process is to add the Google Services library to the dependencies section of the project-level *build.gradle* file (located under *Gradle Scripts -> build.gradle (Project: FirebaseNotify)*):

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0-alpha3'
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
.
.
.
}
```

The second change added the Firebase messaging library and Google Services plugin to the module-level *build.gradle* file (located under *Gradle Scripts -> build.gradle (Module: app)*):

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
.
.
```

```
    dependencies {
        compile 'com.google.firebase:firebase-messaging:10.0.1'
    .
    .
    }
    apply plugin: 'com.google.gms.google-services'
```

With these changes made to the project, the app is ready to begin receiving remote Firebase notifications.

53.9 **Summary**

Firebase provides a range of cloud-based services that can be integrated into web and mobile apps to quickly and easily implement services such as notifications, remote database storage, crash logging and messaging. This chapter has outlined the steps required to create new a Firebase project and to integrate Firebase into an Android Studio project.

54. An Android 7 Firebase Remote Notification Tutorial

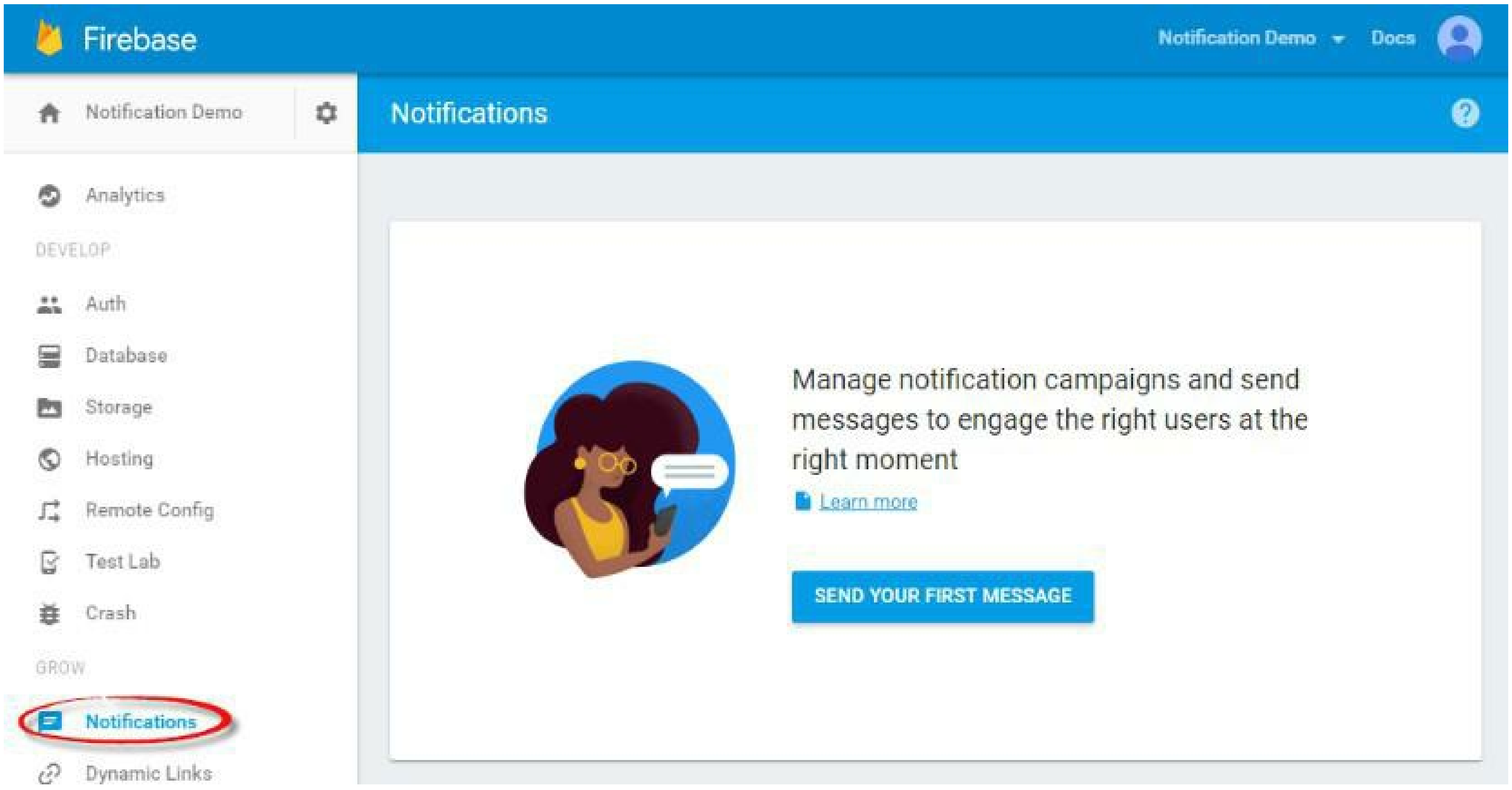
In the preceding chapter a new Firebase project was created and the appropriate steps taken to integrate Firebase into an Android Studio project. These steps were taken in preparation for the Firebase remote notification tutorial covered in this chapter.

This chapter will introduce the Notifications section of the Firebase console and explain how to send a message to a specific app. The app will then be modified to allow the receipt of notifications when the app is in the foreground and to support the inclusion of custom data within the Firebase notification.

54.1 **Sending a Firebase Notification**

Begin by launching the FirebaseNotify app created in the previous chapter on a physical Android device. Once the app is running, place it into the background by tapping the circular home button in the bottom status bar. By default notifications of this type are only delivered to apps that are either not running, or currently in the background.

Open a web browser and navigate to *https://console.firebase.google.com/* to sign into your Firebase console and select the *Firebase Notification Demo* project created in the previous chapter. Once the project has loaded, select the *Notifications* option located in the left-hand panel of the console:

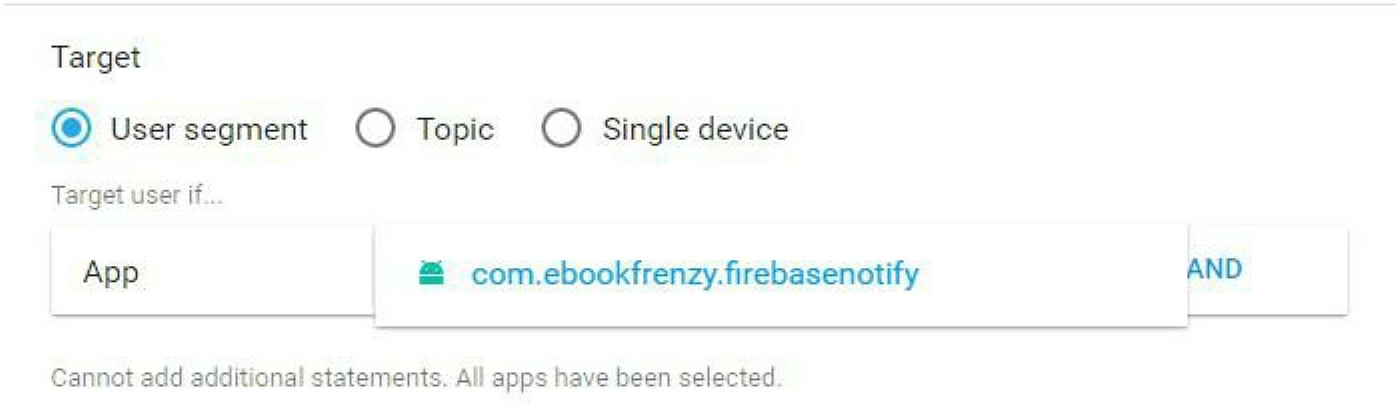


Manage notification campaigns and send messages to engage the right users at the right moment

Figure 54-1

In the main panel, click on the button that reads *Send Your First Message* to display the message composition screen. Enter a text message into the *Message text* field and an optional message label (this is used to reference the notification within the Firebase system and is not seen by the app users) and leave the delivery date to *Send Now*.

In the target section, options are available to target different groups of app users. For the purposes of this example, the notification will target all users of the *com.ebookfrenzy.firebasenotify* app so make sure that *User segment* is selected before choosing the package name from the menu as demonstrated in Figure 54-2:



Target

User segment [✓] · Topic · Single device

Target user if...

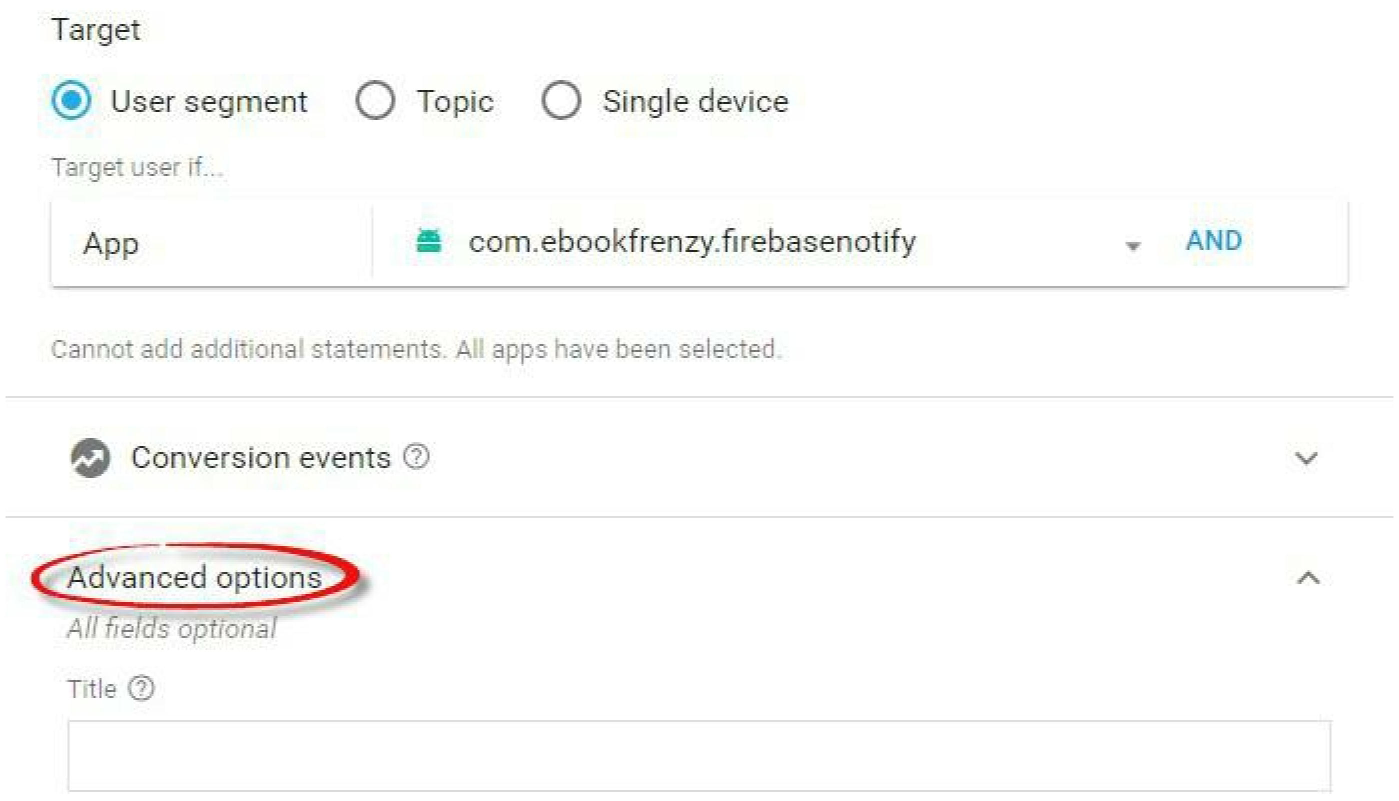
App | com.ebookfrenzy.firebasenotify | AND

Cannot add additional statements. All apps have been selected.

Figure 54-2

Note that the AND option may be used to add additional target criteria such as the version of the app, the spoken language of the user and whether the user has made a previous purchase within the app.

By default, the title of the notification will be set to the name of the corresponding app. The advanced settings section of the Firebase message composition screen allows this title to be changed. To access this setting, click on the *Advanced options* header in the message composition screen as highlighted in Figure 54-3 and enter the title string into the *Title* field:



Target

User segment [✓] · Topic · Single device

Target user if...

App | com.ebookfrenzy.firebasenotify | AND

Cannot add additional statements. All apps have been selected.

Conversion events

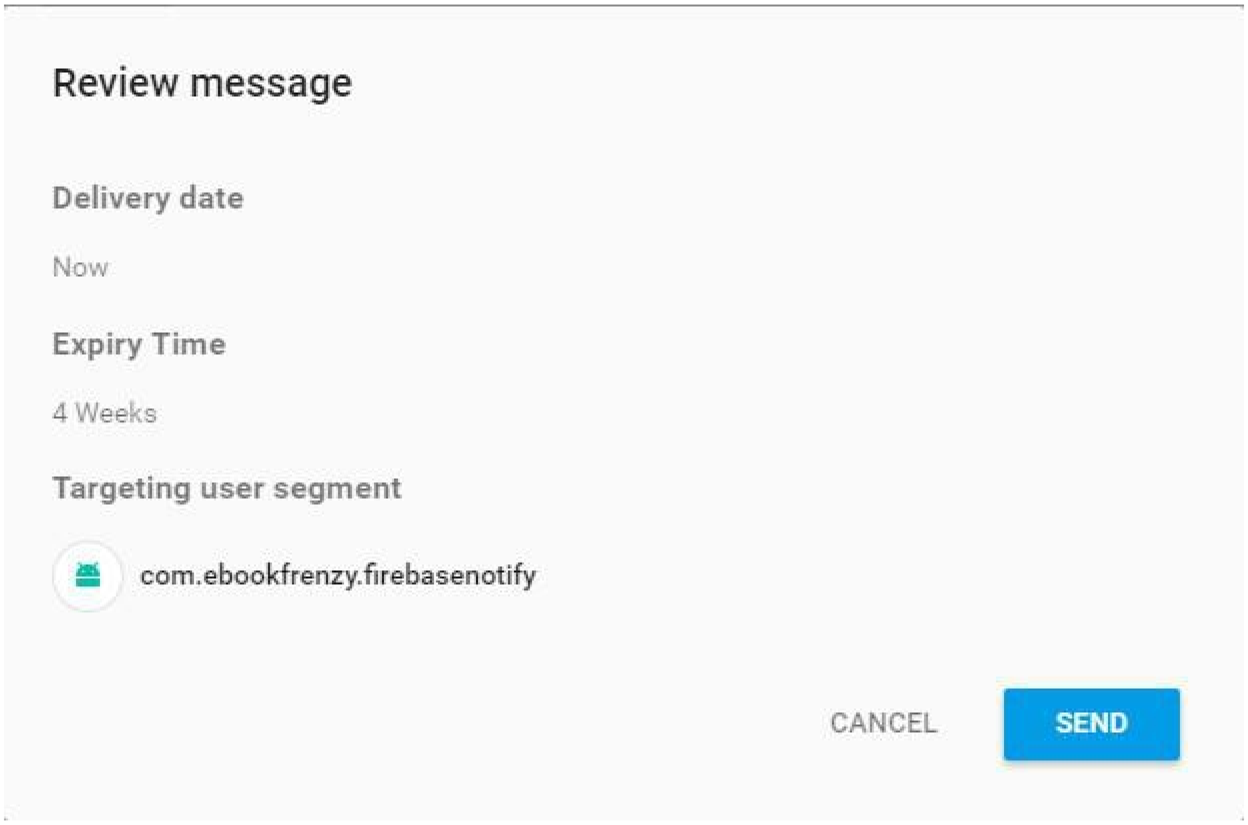
Advanced options

All fields optional

Title

Figure 54-3

With the notification message configured, click on the *Send Message* button, review the settings in the resulting panel and click on the *Send* button:


Review message

Delivery date

Now

Expiry Time

4 Weeks

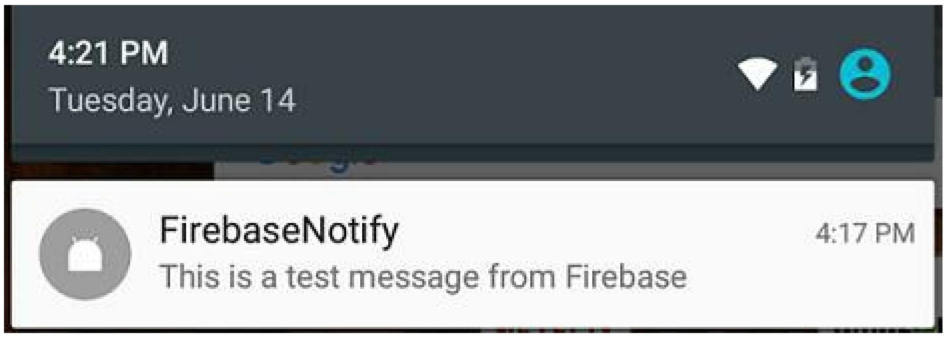
Targeting user segment

com.ebookfrenzy.firebasenotify

Figure 54-4

54.2 Receiving the Notification

After the message has been sent, return to the device on which the app is running and look for a notification indicator in the top status bar. Once the indicator appears, slide downward from the status bar to view the notification which will contain the message text entered when the notification was composed within the Firebase console:


4:21 PM

Tuesday, June 14

FirebaseNotify 4:17 PM

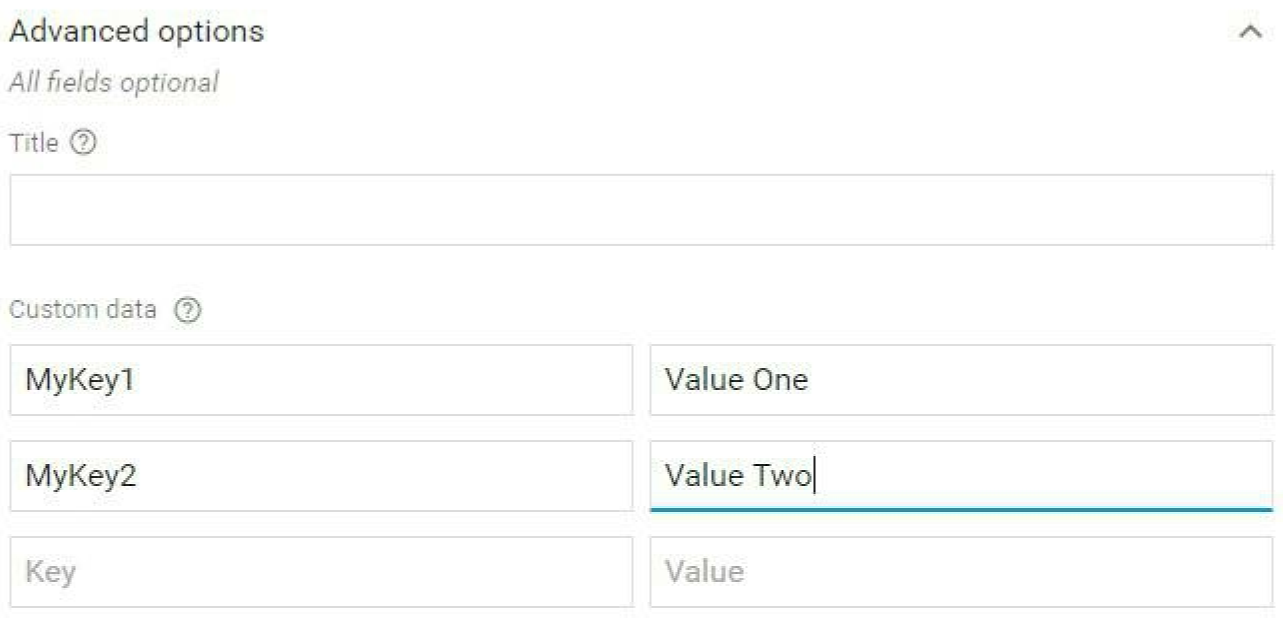
This is a test message from Firebase

Figure 54-5

Tapping the notification will launch the FirebaseNotify app.

54.3 Including Custom Data within the Notification

Firebase messaging also provides the option to pass key-value based data within the notification. This data can then be retrieved by the activity that is launched when the notification is selected on the device. The key-value data pairs to be included with the notification are specified from within the Advanced options section of the message composition screen. Figure 54-6, for example, shows two custom data pairs configured for a notification:



Advanced options

All fields optional

Title

Custom data

MyKey1	Value One
MyKey2	Value Two
Key	Value

Figure 54-6

Within the activity launched when the user taps the notification on the device, the *getIntent()* method may be used to obtain a reference to the Intent object that triggered the launch. Calling the *getExtras()* method on that Intent will return a Bundle object containing the custom data.

The value associated with each key may be accessed by passing through the key value to the *getString()* method of the Bundle object.

Edit the *FirebaseNotifyActivity.java* file and modify the *onCreate()* method to extract the value for a key of "MyKey1" and display that value on the myTextView widget in the activity user interface layout:

```
import android.widget.TextView;
.
.
.
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_firebase_notify);

    Bundle customData = getIntent().getExtras();

    if (customData != null) {

        TextView textView = (TextView) findViewById(R.id.myTextView);
        textView.setText(customData.getString("MyKey1"));
    }
}
```

Build and run the app on a physical Android device and place it into the background. Using the Firebase console, compose a new message targeted at the FirebaseNotify app. Before sending the notification, open the Advanced options panel and enter MyKey1 as the key and a string of your choice as the corresponding value. Send the message, refer to the device on which the app is running and pull down the notification shade when the notification icon appears in the status bar. Tap the notification to launch the activity and note that the string entered into the value field is now displayed on the TextView widget. Custom data has successfully been passed from the Firebase console via a notification to the main activity of the app.

54.4 Foreground App Notification Handling

As previously outlined an app will not, by default, receive a Firebase notification if it is currently the foreground app. In order for a foreground app to receive the notification, it must implement a service that extends the FirebaseMessagingService class and override the *onMessageReceived()* method within that class.

With the FirebaseNotify project loaded into Android Studio, right-click on the *app -> java -> com.ebookfrenzy.firebasenotify* entry and select *New -> Service -> Service* from the menu. In the configuration dialog, name the class *MyFBMessageService* and enable both the *Exported* and *Enabled* options before clicking on the *Finish* button.

Edit the newly created *MyFBMessageService.java* file and modify it to extend the FirebaseMessagingService class, implement the *onMessageReceived()* method and remove the existing *onBind()* method:

```
package com.ebookfrenzy.firebasenotify;

~~import android.app.Service;~~
~~import android.content.Intent;~~
~~import android.os.IBinder;~~
import android.util.Log;

import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFBMessageService extends FirebaseMessagingService  {

    String TAG = "firebasenotify";

    public MyFBMessageService() {
    }

~~    @Override~~
~~    public IBinder onBind(Intent intent) {~~
~~        // TODO: Return the communication channel to the service.~~
~~        throw new UnsupportedOperationException("Not yet implemented");~~
~~    }~~

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        Log.d(TAG, "Notification Title: " +
              remoteMessage.getNotification().getTitle());
        Log.d(TAG, "Notification Message: " +
              remoteMessage.getNotification().getBody());
    }
}
```

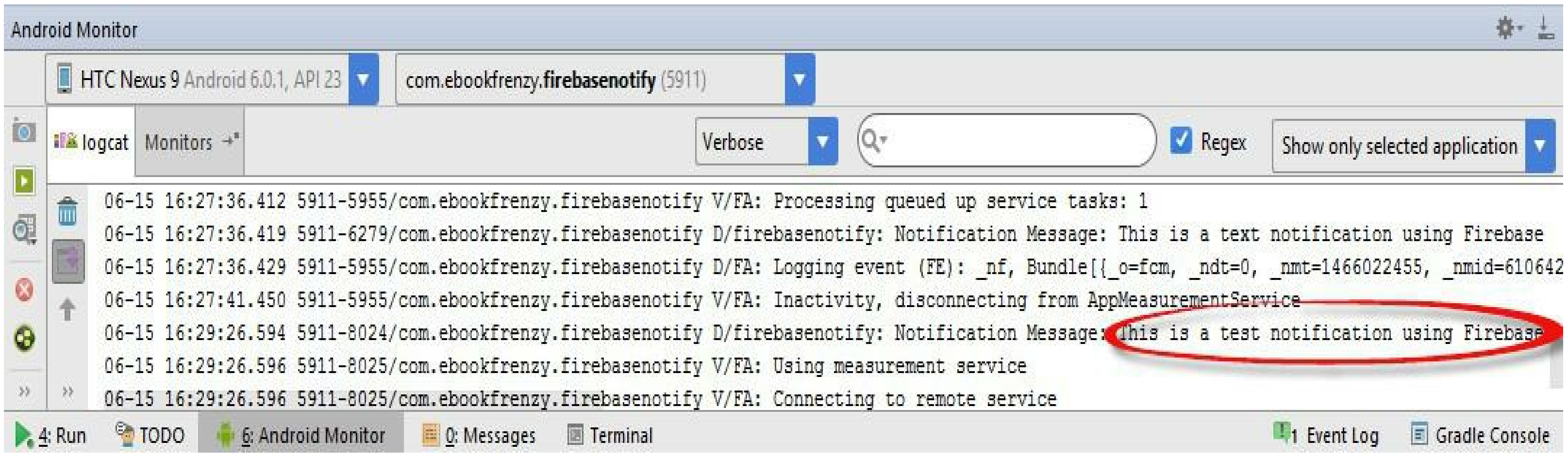
When the *onMessageReceived()* method is called, it is passed as an argument a RemoteMessage object. Contained within this object is an instance of the RemoteMessage.Notification class. This object is used to contain the details of a Firebase remote notification.

In the above code, the *getNotification()* method of the RemoteMessage object is called to access the RemoteMessage.Notification object. The *getTitle()* method of the RemoteMessage.Notification object is then called to obtain the title text of the message while the *getBody()* method returns the notification body text. These strings are both displayed on the Android Studio console.

The final task before testing the code is to modify the service entry within the *AndroidManifest.xml* file to add an intent filter for the Firebase messaging event:

```
<service
    android:name=".MyFBMessageService"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
```

Once the changes have been made, build and run the app on a physical Android device and display the Android Monitor tool window so that the console output from the device is visible. Using the Firebase console, send a new notification to the app using the steps outlined earlier in the chapter. Once the notification has arrived on the device, output should appear in the Android Monitor tool window containing the message title and body text from the notification:



```
06-15 16:27:36.412 5911-5955/com.ebookfrenzy.firebasenotify V/FA: Processing queued up service tasks: 1
06-15 16:27:36.419 5911-6279/com.ebookfrenzy.firebasenotify D/firebasenotify: Notification Message: This is a text notification using Firebase
06-15 16:27:36.429 5911-5955/com.ebookfrenzy.firebasenotify D/FA: Logging event (FE): _nf, Bundle[{_o=fcm, _ndt=0, _nmt=1466022455, _nmid=610642
06-15 16:27:41.450 5911-5955/com.ebookfrenzy.firebasenotify V/FA: Inactivity, disconnecting from AppMeasurementService
06-15 16:29:26.594 5911-8024/com.ebookfrenzy.firebasenotify D/firebasenotify: Notification Message: This is a test notification using Firebase
06-15 16:29:26.596 5911-8025/com.ebookfrenzy.firebasenotify V/FA: Using measurement service
06-15 16:29:26.596 5911-8025/com.ebookfrenzy.firebasenotify V/FA: Connecting to remote service
```

Figure 54-7

54.5 **Summary**

In addition to local notifications, Android also provides a way to send notifications remotely using the Firebase notifications system. Notifications can be targeted to users using a variety of categories including all users of an app, a particular spoken language, an app version or even to a specific Android device. This chapter has demonstrated how to send and receive remote notifications, including the implementation of a service to receive notifications for a foreground app and the passing of custom data to the app from the remote server.