

Abstract

The goal of this document is to function as a developers guide to assist in implementing a webhook service. The service use, for example, is to provide a mechanism that allows a user to receive firebase messages or notifications when a temperature sensor has reached a high or low limit alarm condition during an excursion. This guide illustrates the use of firebase messaging to realize the service. Ultimately 3rd party developers are free to use whatever mechanism they desire in their own code/applications. Please refer to firebase documentation for any troubleshooting or configuration issues.

Adding Firebase Messaging (AKA Webhook) Support

Immediately below are links to the firebase tutorials that will aide you in setting up firebase messaging for your flutter application. Google also offers a [code lab](#) which walks developers through the process.

Important reading materials to guide you through each respective processes, such as:

[Set Up a Flutter Client](#) (platform requirements & [install firebase messaging plugin](#))

[Send a Test Message](#)

[Receive Messages](#)

[Subscribe to Topics](#) (for example, to subscribe to the firebase service that sends data payloads to create notifications)

Register the application for firebase services (in this case, messaging): [Add Firebase to Flutter App](#)

- Install Flutter
- Sign-in to firebase using google account
- Install command line interface (tools)
- Configure apps to use Firebase
- Initialize Firebase in your Application
- Add Firebase Plugins to Dependencies

Auto Generated Files

When using the command line interface to configure your flutter application for firebase messaging, a few files will automatically be adjusted to contain the dependencies in the build.gradle files and 2 files will be generated for you.

These generated files point towards the firebase hosted projects:

firebase_options.dart

google-services.json

Webhook Support for Alarm Notifications

Immediately below are links to the zebra developer portal content that highlight the webhook event subscription API and material that defines/explains webhook functionality.

Documentation for APIs to manage event webhooks subscriptions -

<https://developer.zebra.com/apis/event-subscription-electronic-temperature-sensors>

Zebra Developer Guides:

[Using Webhook Subscriptions](#)

Zebra Blog:

<https://developer.zebra.com/blog/cold-chain-tracking-zebra-electronic-temperature-sensors>

Example Code snippets

Example command to subscribe to all events on a tenant:

POST: <https://api.zebra.com/v2/devices/environmental-sensors/event/subscription/>

headers:

apiKey: <Tenant API Key>

body:

```
{
  "webhookUrl": "<Firebase Function Public URL>",
  "headers":{
    "key":"<Key included with webhook events to verify origin>"
  }
}
```

Example firebase function to trigger a webhook:

```
const {onRequest} = require("firebase-functions/v2/https");
const logger = require("firebase-functions/logger");
const admin = require('firebase-admin');

admin.initializeApp({credential: admin.credential.applicationDefault()});
exports.onEnvirovueDemoWebhookReceived = onRequest((request, response) => {
  var expected_key = ""
  var tenant = request.body._p.analytics.tenant;
  var webhook_key = request.get("key")
  if(expected_key === webhook_key && (tenant === "<Tenant ID>")) {
    var _p = JSON.stringify(request.body._p);
    admin.messaging().sendToTopic("ZSDemo", {"data":{"_p":_p}});
    logger.info("Sent to ZSDemo topic: " + _p);
    response.status(200).send("ok");
    return
  }
  response.status(403).send("Bad Request ");
});
```

Additional Firebase Documentation:

General Best Practices: <https://firebase.google.com/docs/projects/dev-workflows/general-best-practices>

Getting Started on Cloud Functions: <https://firebase.google.com/docs/functions/get-started?gen=2nd>

For the Developer

Aside from the initial setup/configuration the actions to take come down to the following:

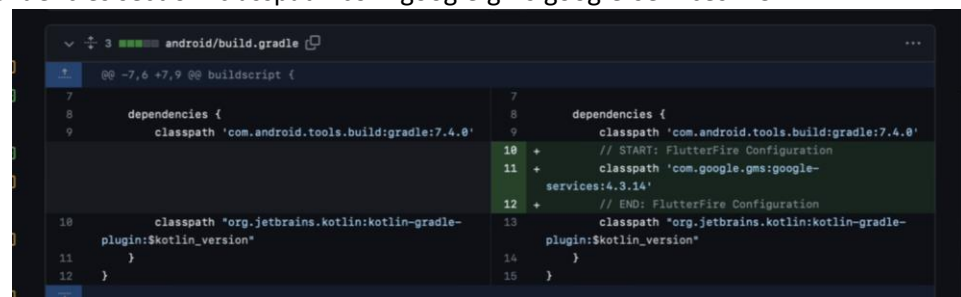
1. Ensuring the correct firebase dependencies have been added to the project. This may vary depending on additional firebase services that may be at play, but for firebase messaging you should at least have the following in your pubspec.yaml file:
firebase_core: ^2.15.1 (version number can vary)
firebase_messaging: ^14.6.6 (version number can vary)
Our example uses flutter_local_notifications plugin to create notifications, so the entry for this would be:
flutter_local_notifications: ^15.1.0+1
2. Ensuring the google-services.json file and firebase_options.dart file received during initial configuration are correct, meaning, the client information points towards to correct firebase project hosted via the [Firebase Console](#). These files are auto generated during the configuration phase.
 - a. google-services.json which contains an application entry for the application that is registering for the firebase messaging event
 - b. firebase_options.dart also mirrors information contained in the google-services.json file, but meant for consumption via the Dart language
3. Verify android/app/build.gradle contains apply-plugin statement: for google services.
 - a. apply plugin: 'com.google.gms.google-services'



The screenshot shows the `android/app/build.gradle` file in an IDE. The file is split into two panes. The left pane shows the original code with lines 22-27. The right pane shows the modified code with lines 22-30. The modification adds the `com.google.gms.google-services` plugin between the `com.android.application` and `kotlin-android` plugins.

```
22 // }
23
24 apply plugin: 'com.android.application'
25
26 apply plugin: 'kotlin-android'
27 apply from:
  "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
28
29
30
```

4. Verify android/build.gradle contains dependency entry for google-services plugin.
 - a. In dependencies section: classpath 'com.google.gms.google-services:4.3.14'



The screenshot shows the `android/build.gradle` file in an IDE. The file is split into two panes. The left pane shows the original code with lines 7-12. The right pane shows the modified code with lines 7-15. The modification adds the `com.google.gms.google-services:4.3.14` classpath dependency to the `dependencies` block.

```
7
8 dependencies {
9   classpath 'com.android.tools.build:gradle:7.4.0'
10
11   classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version'
12 }
13
14
15
```

5. In main.dart file, ensure the firebase services are initialized for the application. Code Snippet:
await Firebase.initializeApp(name: "ExampleAppName", options:
DefaultFirebaseOptions.currentPlatform)
6. In main.dart file, subscribe to firebase message topic. Code Snippet:
await FirebaseMessaging.instance.subscribeToTopic("TopicName");

7. Create and initialize your `FirebaseMessagingService` class which handles receiving of the firebase messages.
 - a. For example, in our `main.dart` file we call `"WebhookNotificationService().init();"`

The following steps are for configuring a firebase function to send the webhook payload:

1. Install firebase CLI tool: <https://firebaseopensource.com/projects/firebase/firebase-tools/#installation>
2. Create firebase function: <https://firebase.google.com/docs/functions/get-started?gen=2nd>
3. Open `functions/index.js` and add code.
4. Run `"firebase login"` to log into firebase.
5. Run `"firebase use --add <project>` to switch to the correct environment.
6. Run `"firebase deploy"` from the root project directory to push the function to google cloud.
7. Subscribe to webhook using the public URL of the new function as the target.

TIPS:

`@Pragma('vm:entry-point')` is required annotation for release build background notification handling on Flutter `>= 3.3.0`. This didn't seem to be required when running a debug build, however, note for release builds this is required to successfully handle background notification (receiving firebase message when the app is not in the foreground)

<https://firebase.google.com/docs/cloud-messaging/flutter/receive> Mentions the `@Pragma` requirement, but also explains more in regards to receiving firebase messages in flutter in general.

Ensure there is a default image for the notification that will be created based on the firebase message. In our experience we have noticed without a default image for the notification, it may never show up. Behavior seemed to be dependent on device manufacturer.

Additional General Documentation :

Overview of firebase messaging: <https://firebase.flutter.dev/docs/messaging/overview/>

Introduction of cloud messaging: <https://firebase.google.com/docs/cloud-messaging>

Source Code Examples:

View `lib/services/webhook_notification_service.dart` for an example implementation for a service that receives firebase messages and creates notifications in response.

View `lib/main.dart` for an illustration of how to initialize firebase services & webhook subscriptions (referred to in 'item 5' and 'item 6' in developer TLDR).