



Firebase Remote Config

Firebase Remote Config is a cloud service that lets you change the behavior and appearance of your app without requiring users to download an app update. When using Remote Config, you create in-app default values that control the behavior and appearance of your app. Then, you can later use the Firebase console or the Remote Config backend APIs to override in-app default values for all app users or for segments of your user base. Your app controls when updates are applied, and it can frequently check for updates and apply them with a negligible impact on performance.

Check the [official page](#) for more information.

Setup

Before you can start using any Firebase module extensions you are required to follow some initial configuration. However if you've already done these for any of the other modules you can skip this configuration section and go straight to using the API functions.

- [Create Project](#)
- [Platform Setup](#) (Remote Config enabling steps are required)

Functions

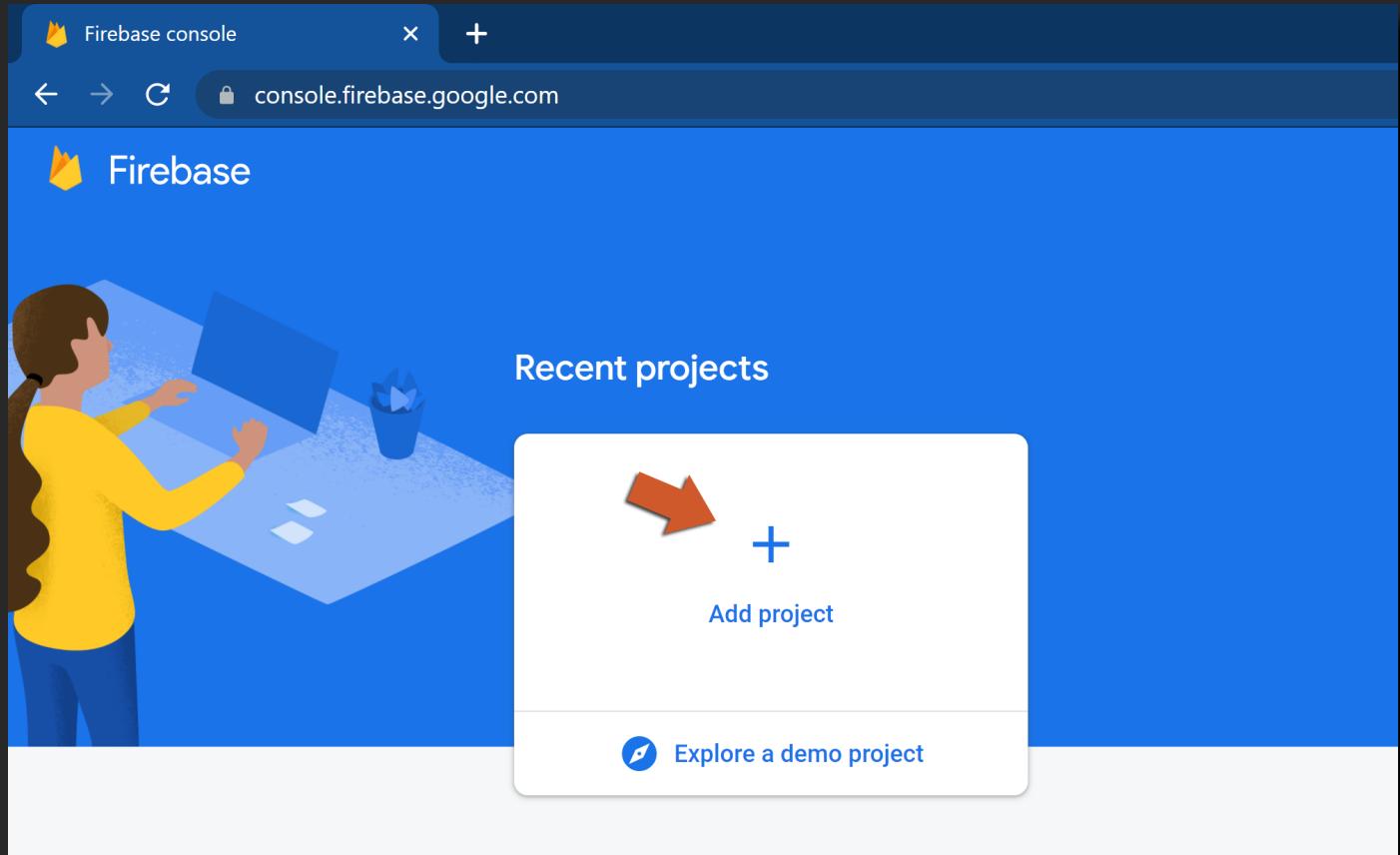
The following functions are given for working with Firebase Remote Config extension:

- [FirebaseRemoteConfig_FetchAndActivate](#)
- [FirebaseRemoteConfig_GetDouble](#)
- [FirebaseRemoteConfig_GetKeys](#)
- [FirebaseRemoteConfig_GetString](#)

Create Project

Before working with any Firebase functions, you must set up your Firebase project:

1. Go to the [Firebase Console](#) web site.
2. Click on **Add Project** to create your new project.



3. Enter a name for your project and click on the **Continue** button.

- ✗ Create a project (Step 1 of 3)

Let's start with a name for
your project?

Enter your project name

my-awesome-project-id

Select parent resource

Continue

4. On the next page, make sure that **Enable Google Analytics for this project** is enabled and then click the **Continue** button:

- ✗ Create a project (Step 2 of 3)

Google Analytics

for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.

Google Analytics enables:

💡 A/B testing ?

🌀 Crash-free users ?

🌐 User segmentation & targeting across
Firebase products

👉 Event-based Cloud Functions triggers ?

📈 Predicting user behavior ?

📊 Free unlimited reporting ?



Enable Google Analytics for this project
Recommended

Previous



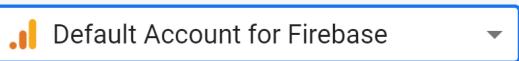
Continue

5. Select your account and click the **Create project** button:

X Create a project (Step 3 of 3)

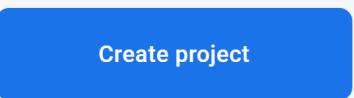
Configure Google Analytics

Choose or create a Google Analytics account [?](#)

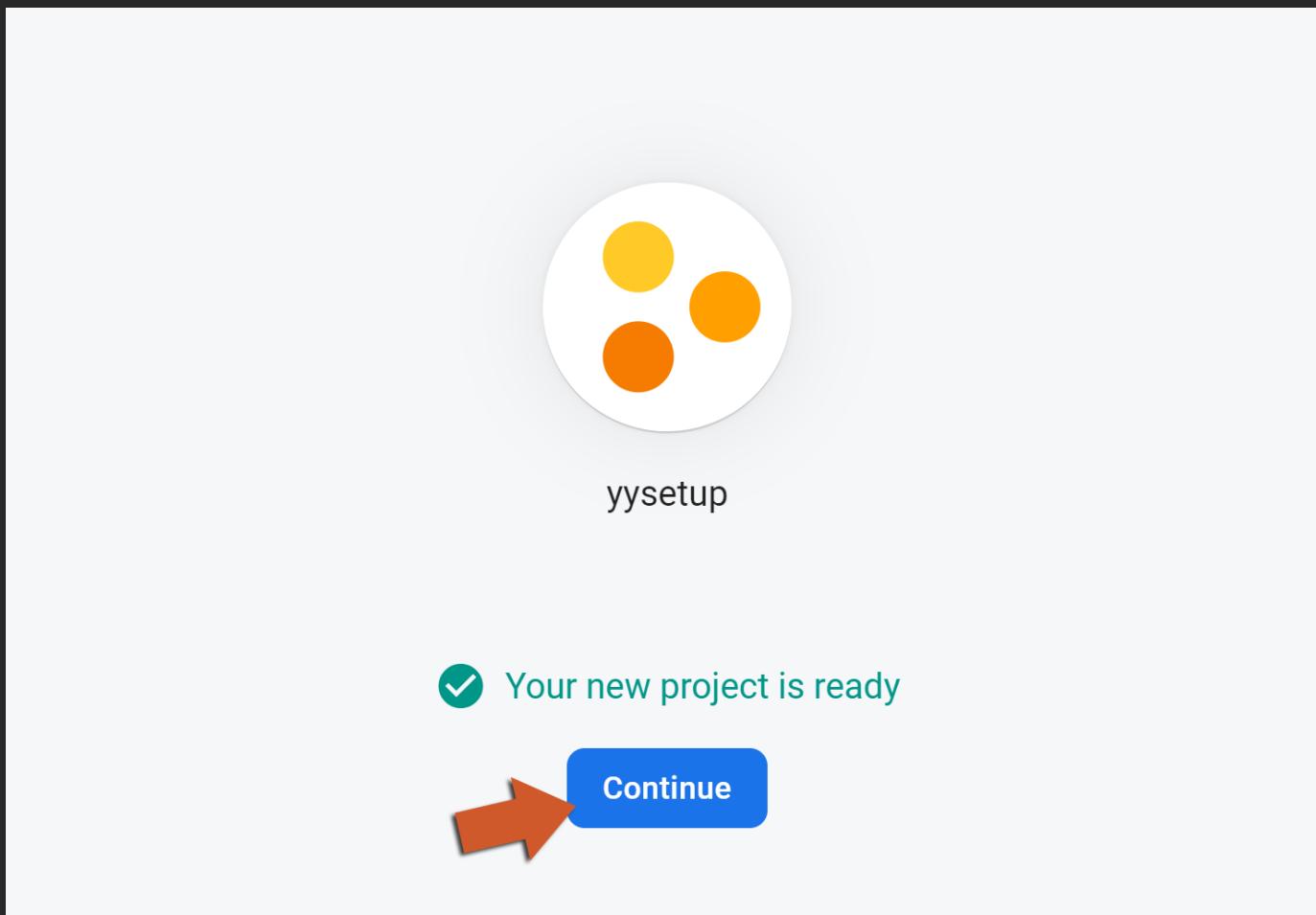
 

Automatically create a new property in this account 

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more.](#)

[Previous](#)  

6. Wait a moment until your project is created; after a few moments you should see the page shown below:



7. You will now be taken to your new project's home page:

The screenshot shows the Firebase console interface. On the left, a dark sidebar lists various services: Project Overview, Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning, Crashlytics, Performance, Test Lab, App Distribution, and Extensions. Below the sidebar, it says 'Spark Free \$0/month' and has an 'Upgrade' button. The main area is titled 'ysetup' with a 'Spark plan' button. It features a large blue background image of two people interacting with a smartphone. Text in the center reads 'Get started by adding Firebase to your app' and 'Add an app to get started'. Below this, there are icons for iOS, Android, code, and a speaker. A modal window at the bottom is titled 'Store and sync app data in milliseconds' and shows two colorful icons related to data storage.

8. Continue your adventure with the Firebase extensions provided for GameMaker!

Platform Setup

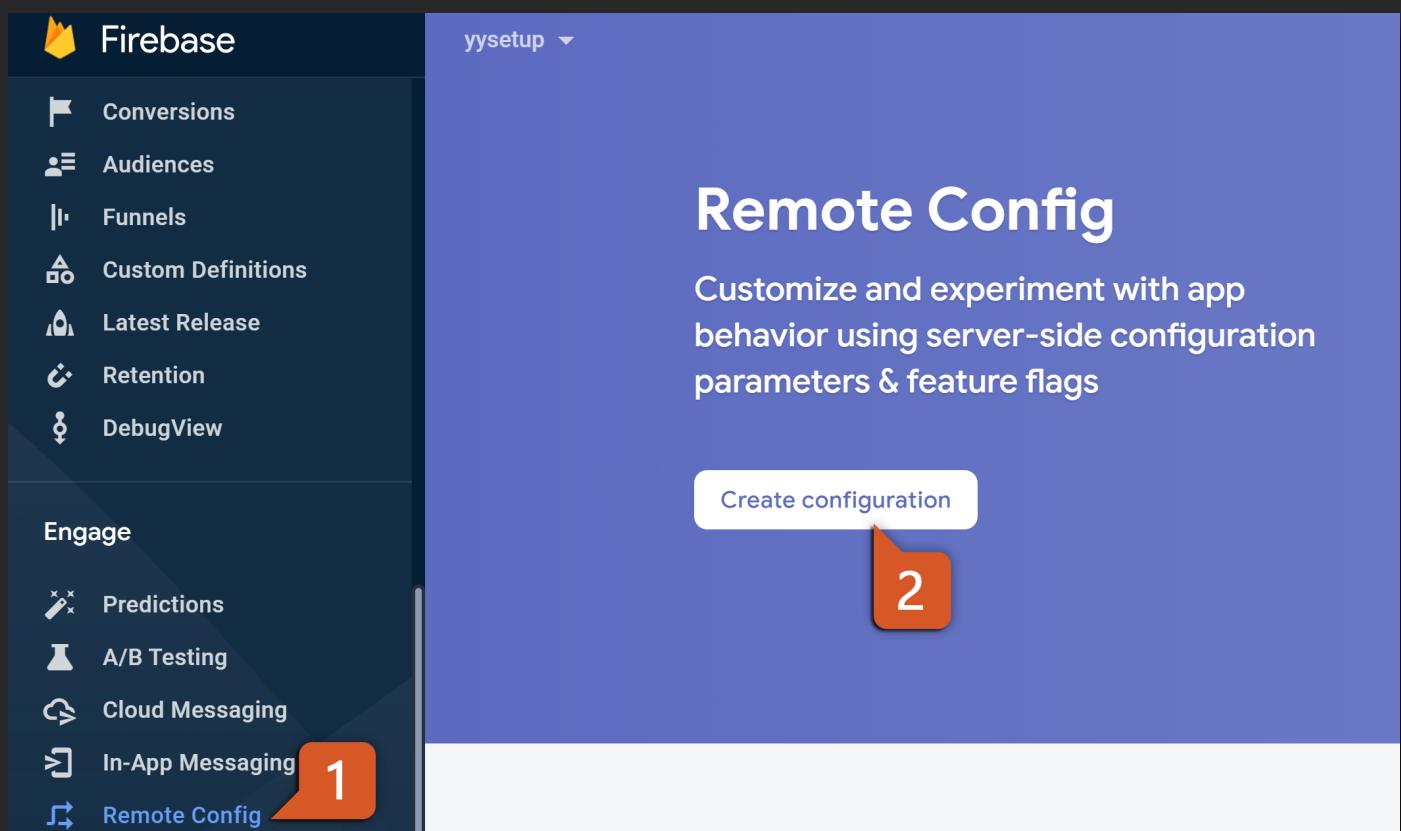
The Firebase Remote Config implementation uses SDK dependencies and therefore is only available on **Android**, **iOS** and **Web** targets. In this section we will cover the required setup necessary to start using the Firebase Remote Config extension in your game.

Select your target platform below and follow the simple steps to get your project up and running (you only need follow these topics once per project):

- [Android Setup](#)
- [iOS Setup](#)
- [Web Setup](#)

Enabling Remote Config

1. Go to **Remote Config** and click on the **Create configuration** button:



2. Click on the **Add parameter** button:

Name	Condition	Value	Fetch %	Actions
ABC data1	Default value	Opera		
ABC data0	Default value	YoYoGames	0%	

3. Add a Parameter name (key) and a Default value (string/real), then click Save:

Create parameter X

Parameter name (key) ②

Description ①

Optional

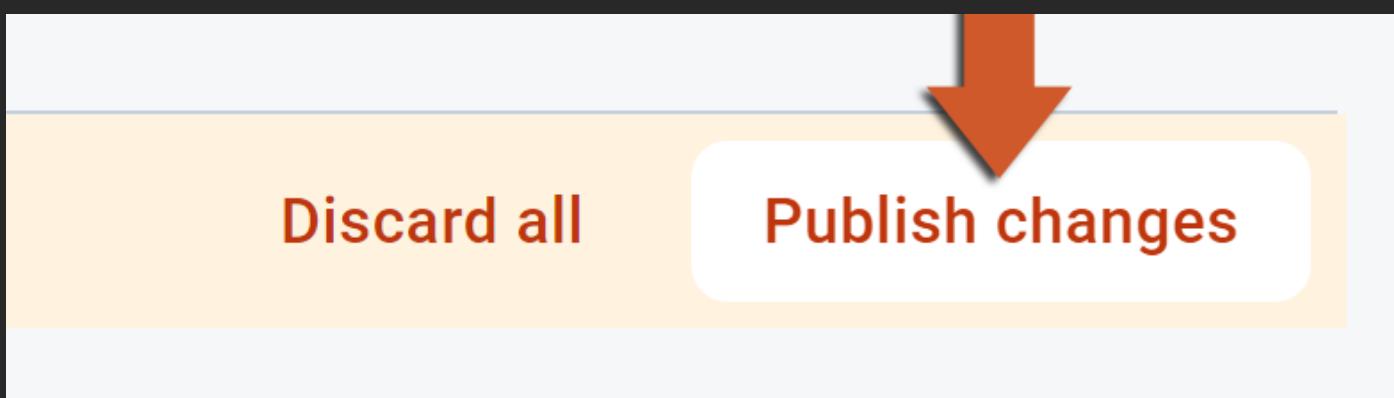
Data type ABC String ②

Default value ③

Use in-app default

+ Add new ▾ Cancel Save

4. After adding all your parameters, don't forget to click Publish changes:

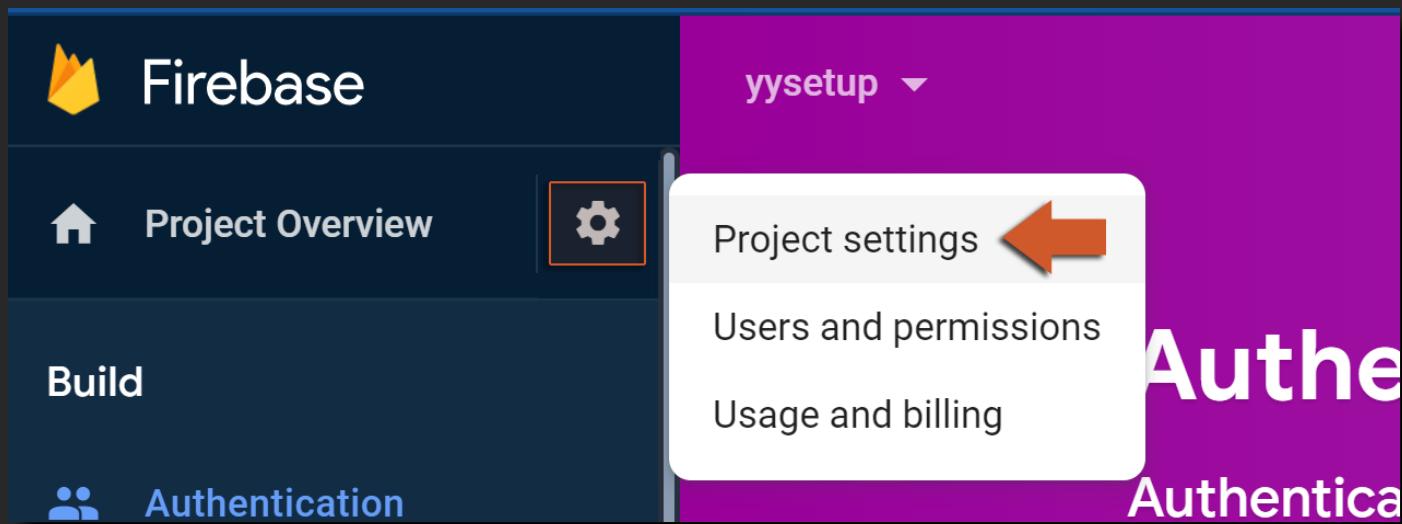


Android Setup

This setup is necessary for all the Firebase modules using Android and needs to be done once per project, and basically involves importing the `google-services.json` file into your project.

IMPORTANT Please refer to [this Helpdesk article](#) for instructions on setting up an Android project.

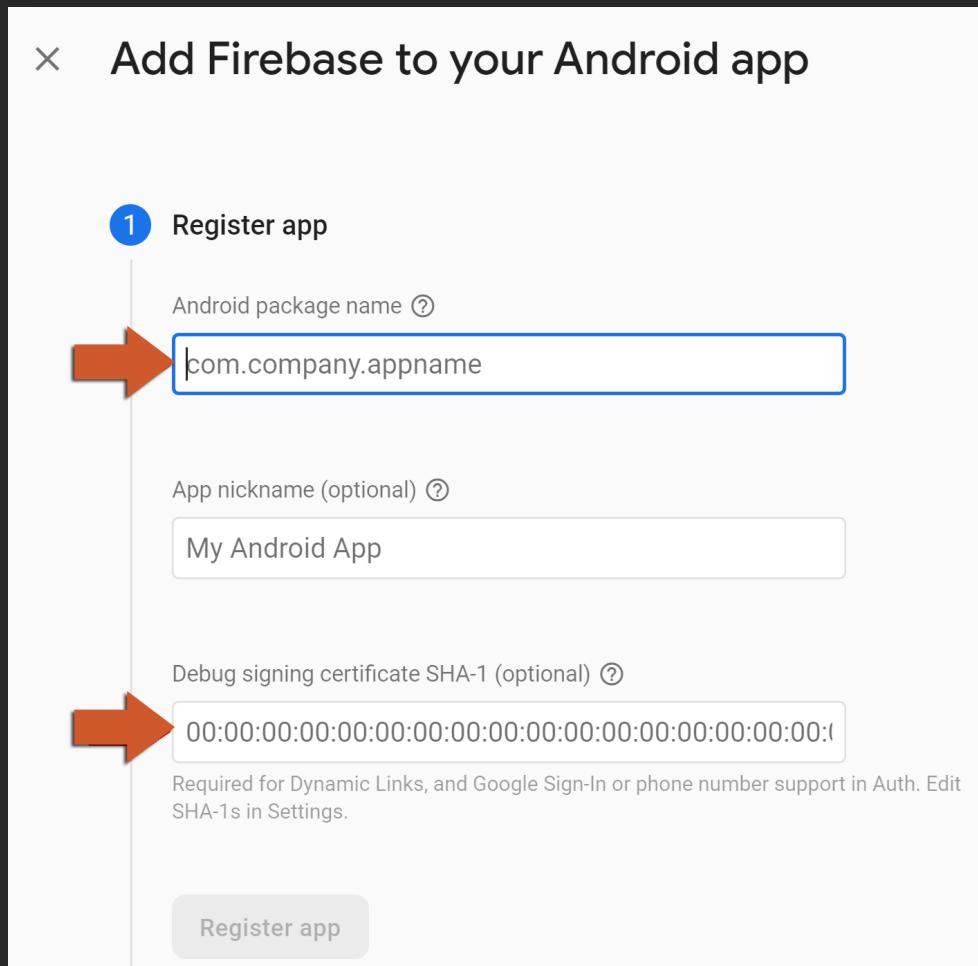
1. Click the **Settings** icon (next to Project Overview) and select **Project settings**:



2. Now go to the **Your apps** section and click on the **Android** button:

The screenshot shows the 'Project settings' page under the 'ysetup' dropdown. The 'Your apps' section is visible, showing a message: 'There are no apps in your project. Select a platform to get started.' Below this message are four circular icons representing different platforms: iOS, TV, Android (highlighted with a red box), and Web. The 'Your apps' button is also highlighted with a red box.

3. On this screen you need enter your **Package name** (required), **App nickname** (optional) and **Debug signing certificate SHA-1** (required if you are using Firebase Authentication).



You can get your package name from the [Android Game Options](#), and your **Debug signing certificate SHA-1** from the [Android Preferences](#) (under Keystore):



4. Click on **Download google-services.json** (make sure to save this file as we will need it in subsequent steps).

× Add Firebase to your Android app

✓ Register app

Android package name: com.yoyogames.YoyoPlayServices2, App nickname: yysetup_android

2 Download config file

Instructions for Android Studio below | [Unity](#) [C++](#)

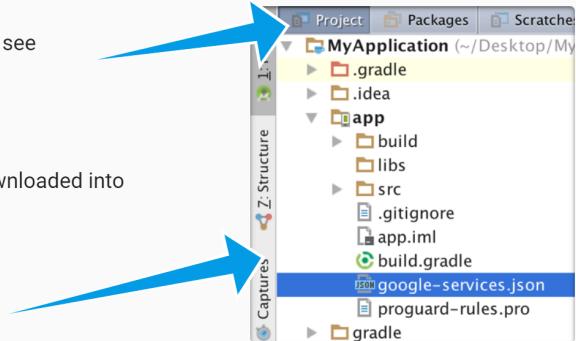
 [Download google-services.json](#)

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.



google-services.json



[Next](#)

5. Ignore this screen, as this is already done in the extension.

× Add Firebase to your Android app

✓ Register app

Android package name: com.yoyogames.YoyoPlayServices2, App nickname: yysetup_android

Download config file

3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

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 {  
        ...  
    }  
}
```

6. Click on the Continue to console button.

× Add Firebase to your Android app

Register app

Android package name: com.yoyogames.YoyoPlayServices2, App nickname: yysetup_android

Download config file

Add Firebase SDK

4 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

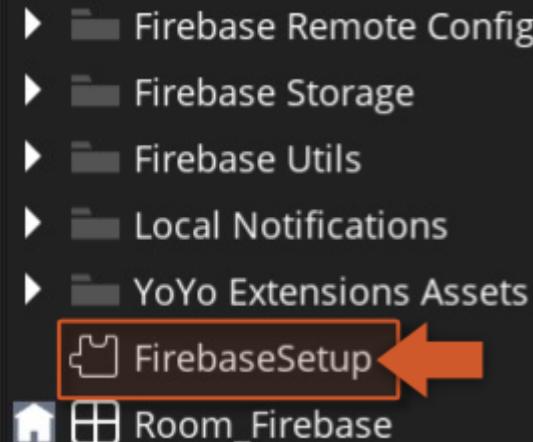
You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

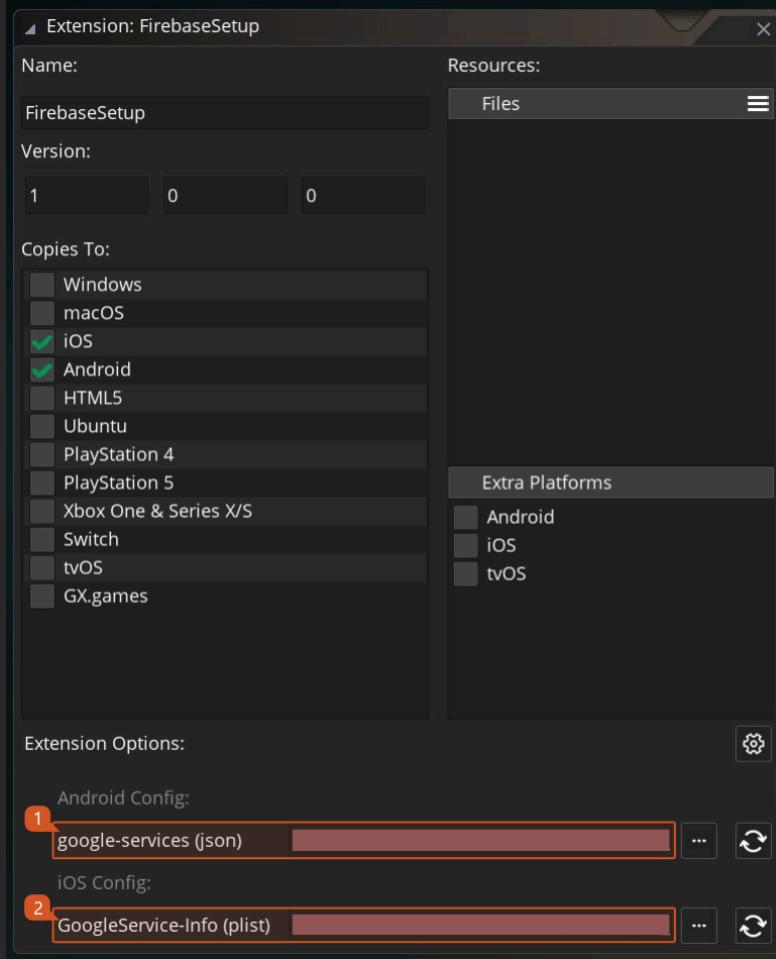
Previous

Continue to console

7. Now go into GameMaker, double click the extension **FirebaseSetup** asset.



8. In the extension panel just fill in the paths for the correct files (Android and/or iOS).



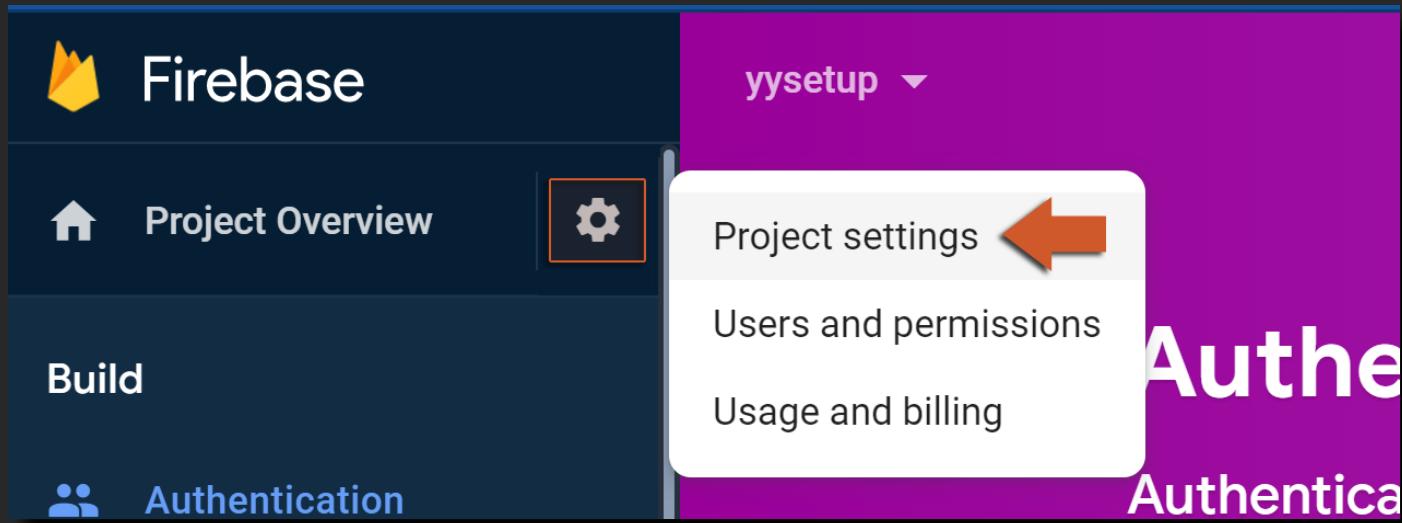
9. You have now finished the main setup for all Firebase Android modules!

iOS Setup

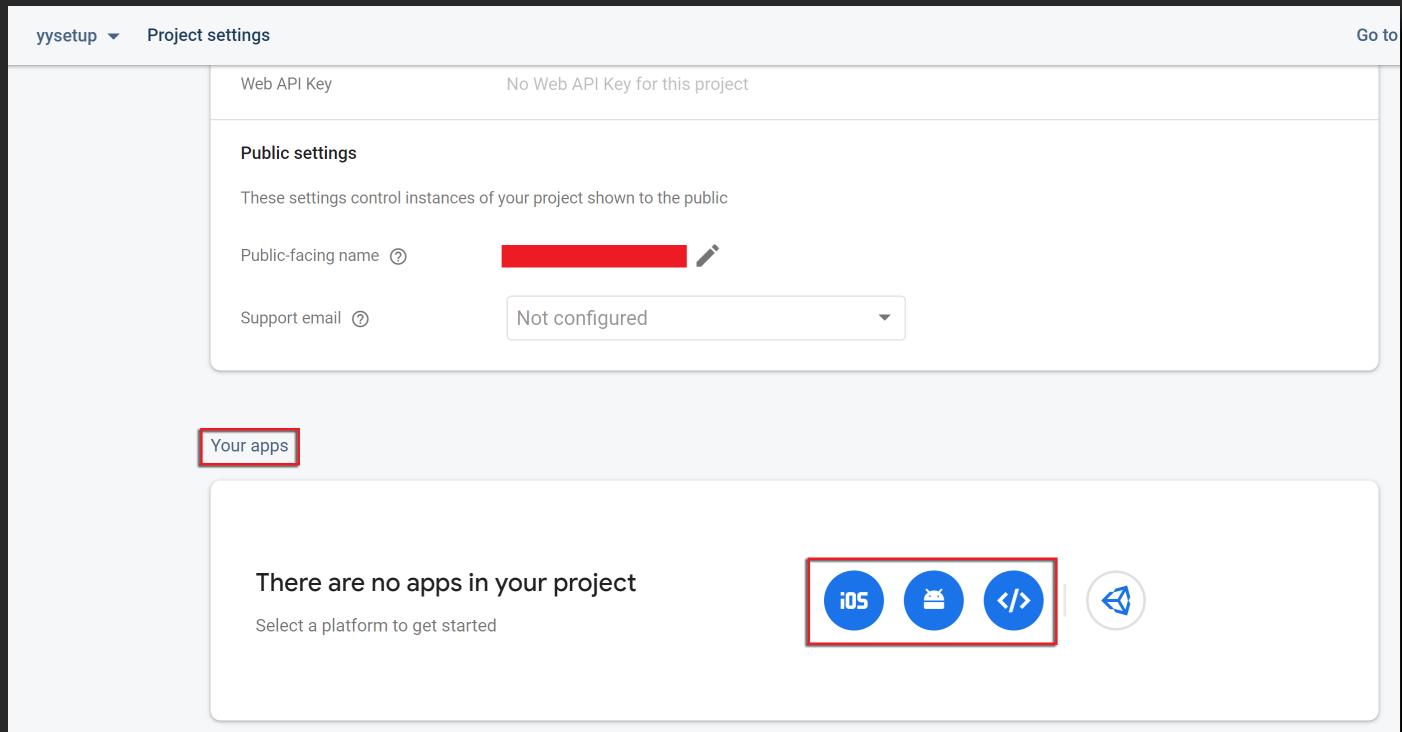
This setup is necessary for all the Firebase modules using iOS and needs to be done once per project, and basically involves importing the `GoogleServices-Info.plist` file into your project.

IMPORTANT Please refer to [this Helpdesk article](#) for instructions on setting up an iOS project.

1. Click the **Settings** icon (next to Project Overview) and select **Project settings**:



2. Now go to the **Your apps** section and click on the **iOS** button:



3. Fill the form with your iOS Bundle ID, App nickname and AppStore ID (last two are optional).

X Add Firebase to your iOS app

1 Register app

iOS bundle ID ?



App nickname (optional) ?

App Store ID (optional) ?

4. Click on **Download GoogleService-info.plist** (make sure to save this file as we will need it in subsequent steps).

× Add Firebase to your iOS app

✓ Register app

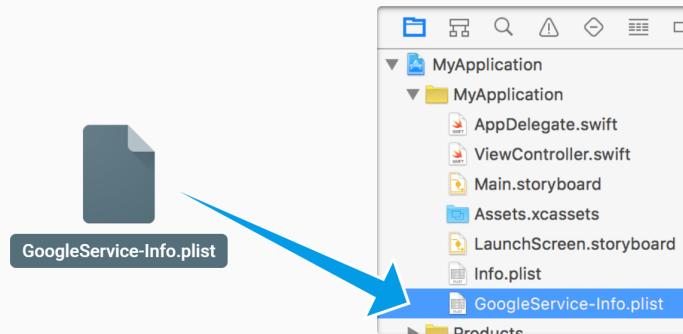
iOS bundle ID: com.yoyogames.yyfirebase

2 Download config file

Instructions for Xcode below | [Unity](#) [C++](#)

→ [Download GoogleService-Info.plist](#)

Move the GoogleService-Info.plist file you just downloaded into the root of your Xcode project and add it to all targets.



[Next](#)

5. Ignore this screen, as this is already done in the extension.

× Add Firebase to your iOS app

✓ Register app

iOS bundle ID: com.yoyogames.yyfirebase

Download config file

3 Add Firebase SDK

Instructions for CocoaPods | [SwiftPM](#) [Download ZIP](#) [Unity](#) [C++](#)

Google services use [CocoaPods](#) to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don't have one:

\$ pod init



Open your Podfile and add:

```
# add the Firebase pod for Google Analytics
```

6. Ignore this screen as well, as this is also done in the extension.

× Add Firebase to your iOS app

- ✓ Register app
iOS bundle ID: com.yoyogames.yyfirebase

- Download config file

- Add Firebase SDK

4 Add initialization code

To connect Firebase when your app starts up, add the initialization code below to your main `AppDelegate` class.

Swift Objective-C

```
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
```



7. Click on the Continue to console button:

× Add Firebase to your iOS app

- ✓ Register app
iOS bundle ID: com.yoyogames.yyfirebase

- Download config file

- Add Firebase SDK

- Add initialization code

5 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

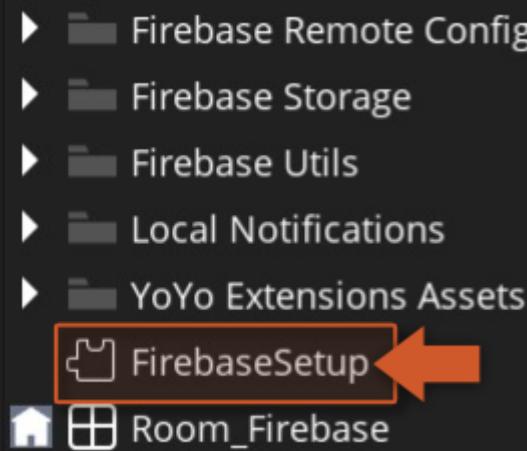
Or, continue to the console to explore Firebase.

Previous

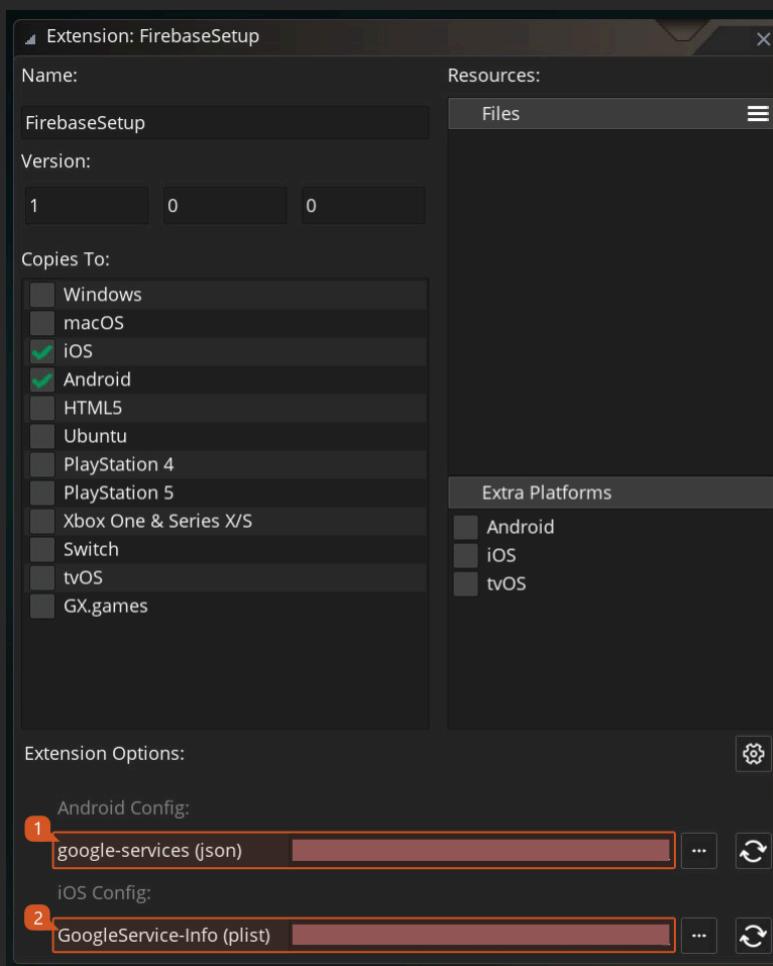
Continue to console



8. Now go into GameMaker, double click the extension **FirebaseSetup** asset.



9. In the extension panel just fill in the paths for the correct files (Android and/or iOS).



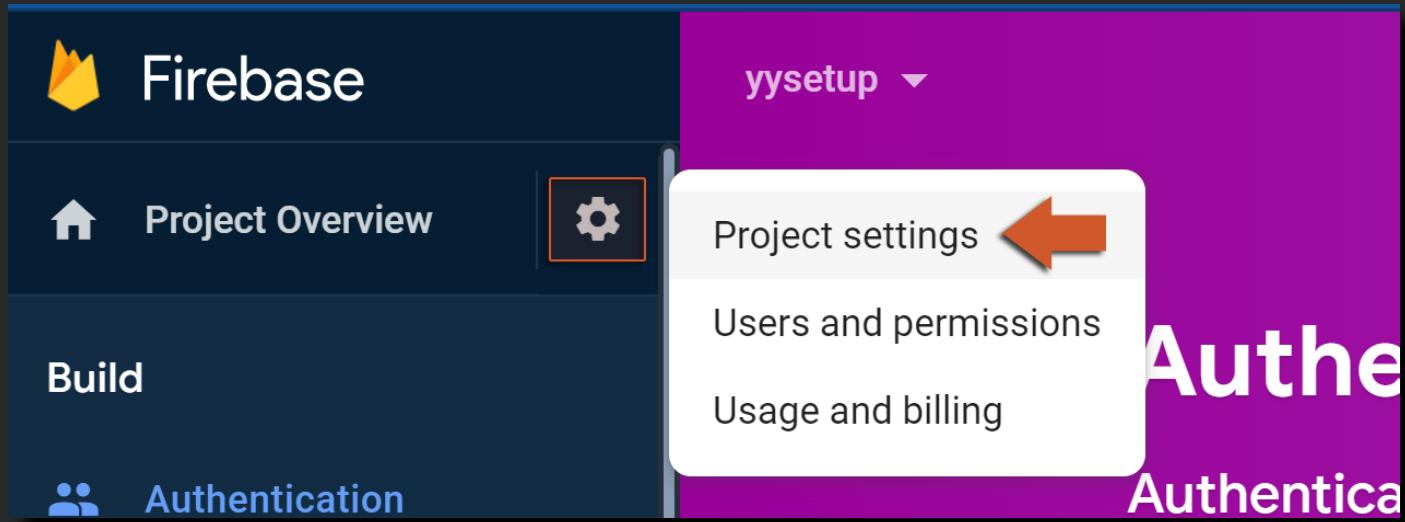
10. Make sure to set up **CocoaPods** for your project *unless* you are only using the REST API in an extension (if one is provided -- not all extensions provide a REST API) or the Firebase Cloud Functions extension (which only uses a REST API).

11. You have now finished the main setup for all Firebase iOS modules!

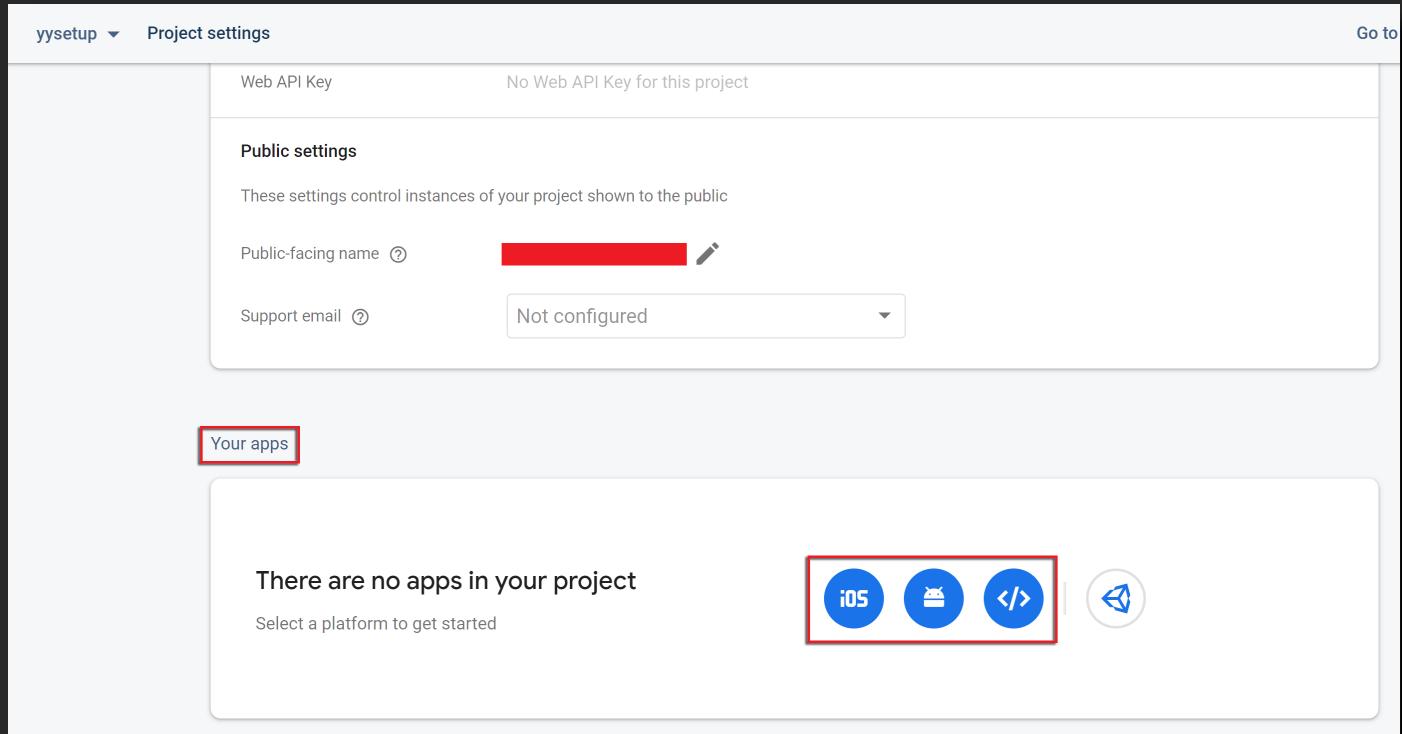
Web Setup

This setup is necessary for all the Firebase modules using Web export and needs to be done once per project, and basically involves adding Firebase libraries and your Firebase values to the `index.html` file in your project.

1. Click the **Settings** icon (next to Project Overview) and then **Project settings**:



2. Now go to the **Your apps** section and click on the **Web (</>)** button:



3. Enter your App nickname (required):

X Add Firebase to your web app

1 Register app

App nickname ⓘ

My web app



! An app nickname is required.

Also set up **Firebase Hosting** for this app. [Learn more ↗](#)

Hosting can also be set up later. It's free to get started anytime.

Register app

4. On this screen, just copy the `firebaseConfig` struct:

2 Add Firebase SDK

Use npm ⓘ Use a <script> tag ⓘ

If you're already using [npm ↗](#) and a module bundler such as [webpack ↗](#) or [Rollup ↗](#), you can run the following command to install the latest SDK:

```
$ npm install firebase
```



Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries
```

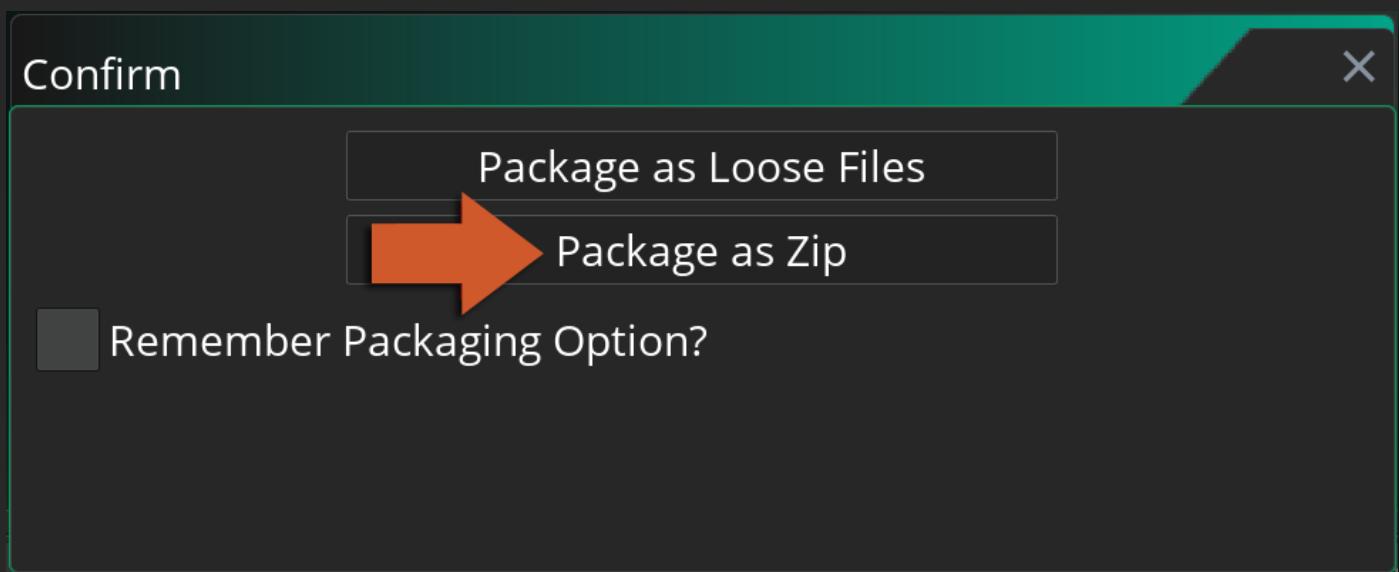
```
// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: [REDACTED],
  authDomain: [REDACTED],
  projectId: [REDACTED],
  storageBucket: [REDACTED],
  messagingSenderId: [REDACTED],
  appId: [REDACTED],
  measurementId: [REDACTED]
};
```

```
// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

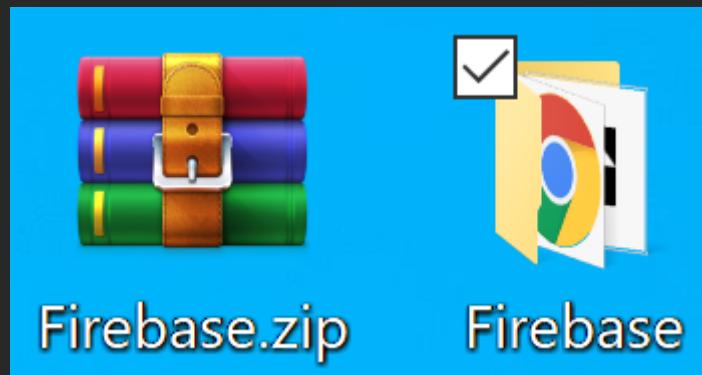


5. Now go back into GameMaker Studio 2 and build your project.

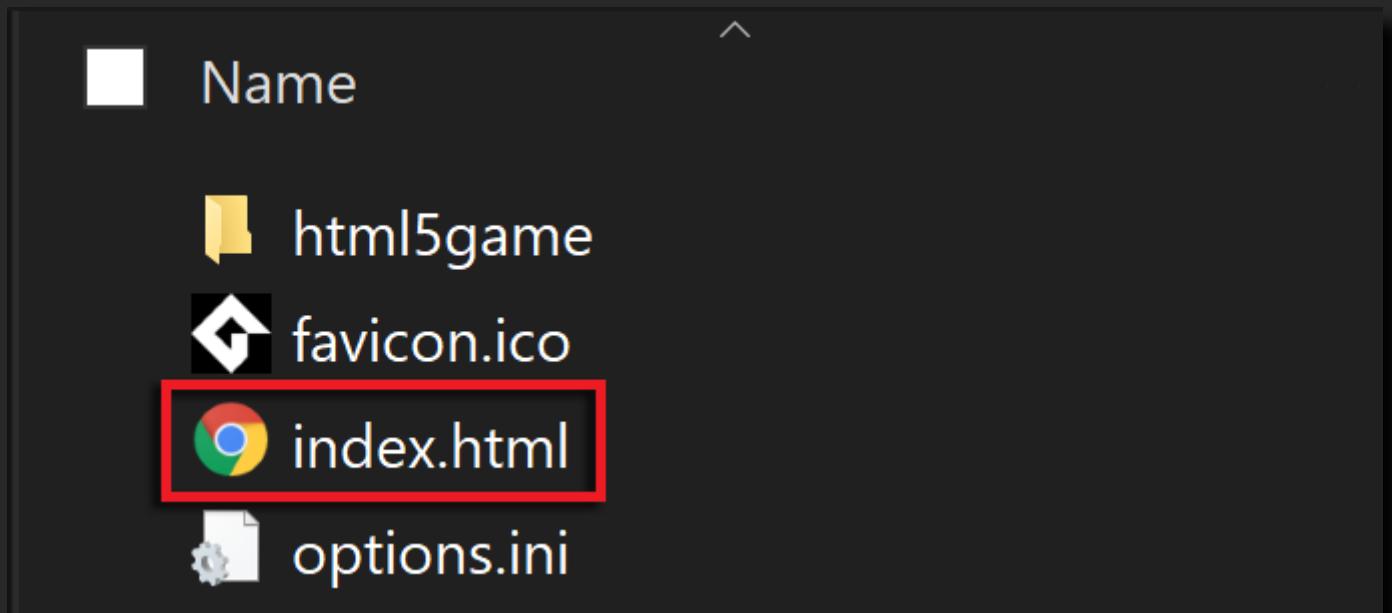
6. Choose the **Package As Zip** option:



7. Locate the created package and **extract it** into a folder.



8. Open the extracted folder and look for an **index.html** file.



9. Open the **index.html** file in Notepad++ or Visual Studio Code (or any other text editor you prefer).

```

65  /* END - Login Dialog Box */
66  :webkit-full-screen {
67      width: 100%;
68      height: 100%;
69  }
70  </style>
71  </head>
72
73 <body>
74     <div class="gm4html5_div_class" id="gm4html5_div_id">
75         <!-- Create the canvas element the game draws to -->
76         <canvas id="canvas" width="1366" height="768" >
77             <p>Your browser doesn't support HTML5 canvas.</p>
78         </canvas>
79     </div>
80
81     <!-- Run the game code -->

```

10. Now we need to add the following code between the `</head>` and `<body>` tags (line 72 in the `html.index` image above):

```

<script src="https://www.gstatic.com/firebasejs/8.9.1.firebaseio.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseAnalytics.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseAuth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1.firebaseio.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseRestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.9.1/firebaseRemoteConfig.js"></script>

<script>
  const firebaseConfig = {
    apiKey: "",
    authDomain: "",
    databaseURL: "",
    projectId: "",
    storageBucket: "",
    messagingSenderId: "",
    appId: "",
    measurementId: ""
  };
  firebase.initializeApp(firebaseConfig);

</script>

```

11. Replace the `const firebaseConfig` part with the one copied in step 4:

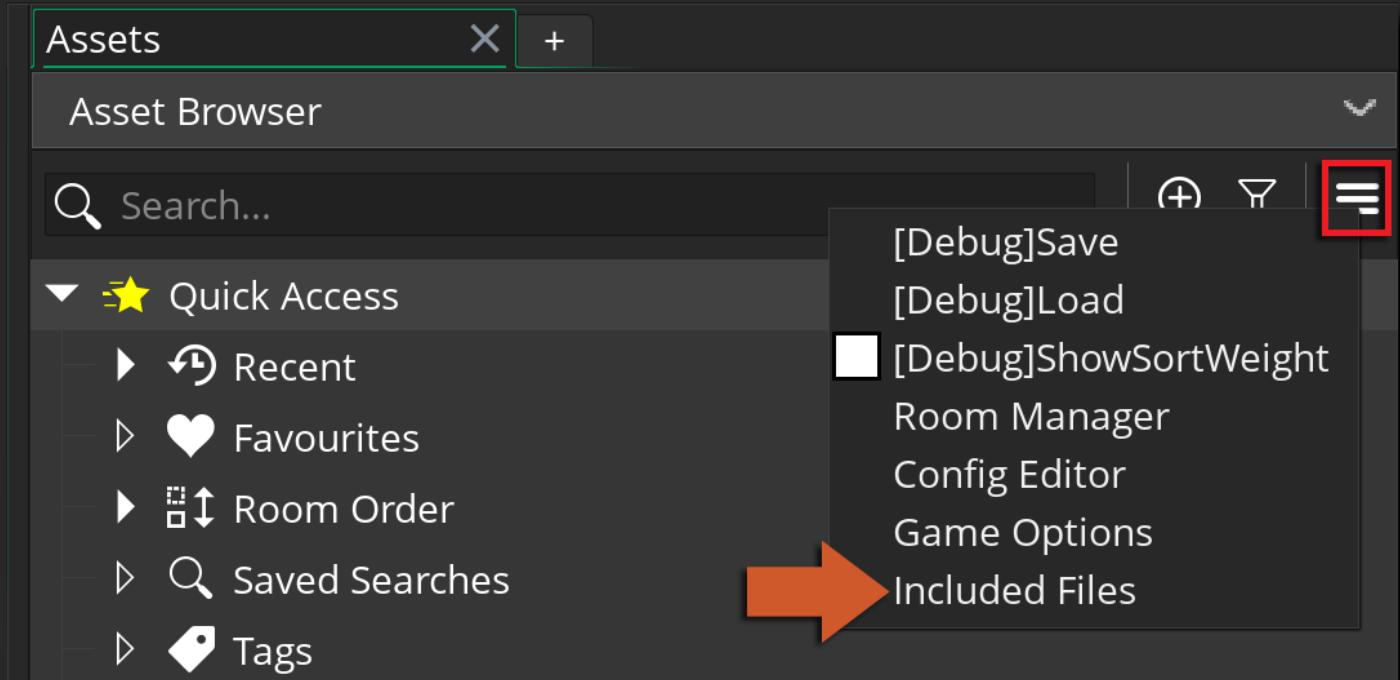
```

68         height: 100%;
69     }
70 
```

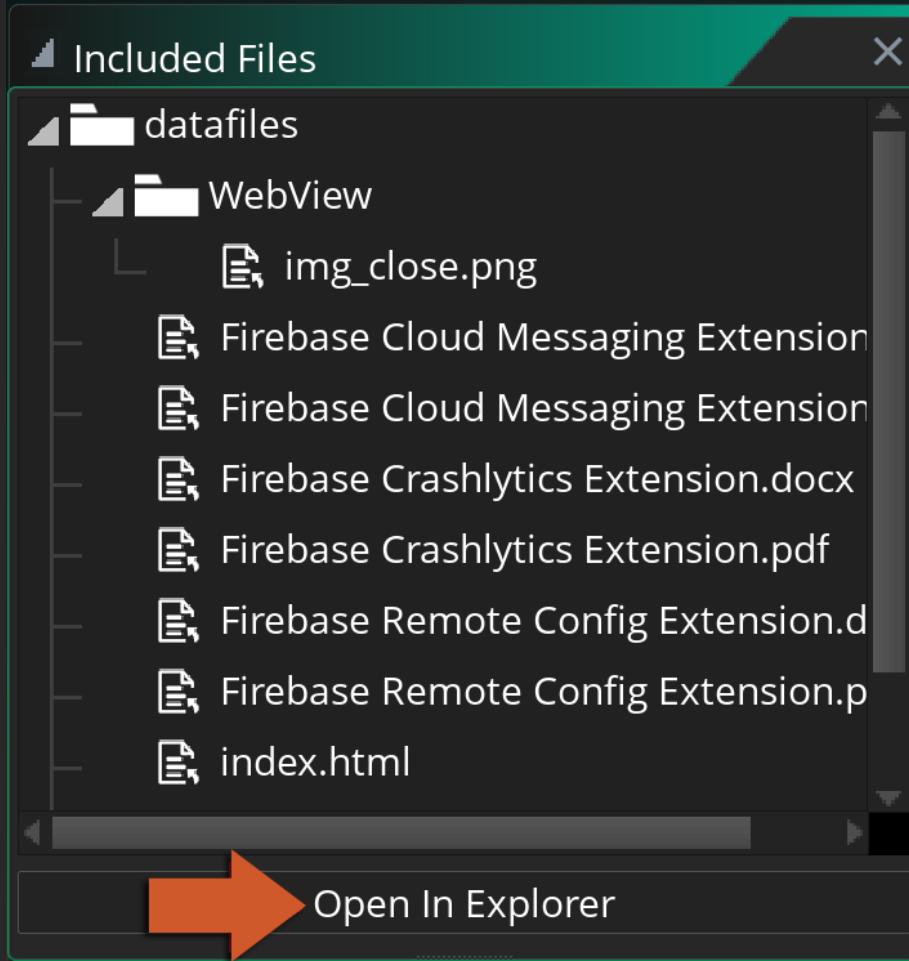
```

71 </style>
72 </head>
73
74 <!-- Firebase -->
75 <script src="https://www.gstatic.com/firebasejs/8.9.1.firebaseio.js"></script>
76 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-analytics.js"></script>
77 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-auth.js"></script>
78 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-database.js"></script>
79 <script src="https://www.gstatic.com/firebasejs/8.9.1/firebase-firebase.js"></script>
80
81
82 <script>
83
84 // Your web app's Firebase configuration
85 // For Firebase JS SDK v7.20.0 and later, measurementId is optional
86 const firebaseConfig = {
87   apiKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
88   authDomain: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
89   databaseURL: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
90   projectId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
91   storageBucket: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
92   messagingSenderId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
93   appId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
94   measurementId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
95 };
96
97 firebase.initializeApp(firebaseConfig);
98
99 </script>
100
101
102
103 <body>
104     <div class="gm4html5_div_class" id="gm4html5_div_id">
```

12. Go back into GameMaker and open your **Included Files** folder.



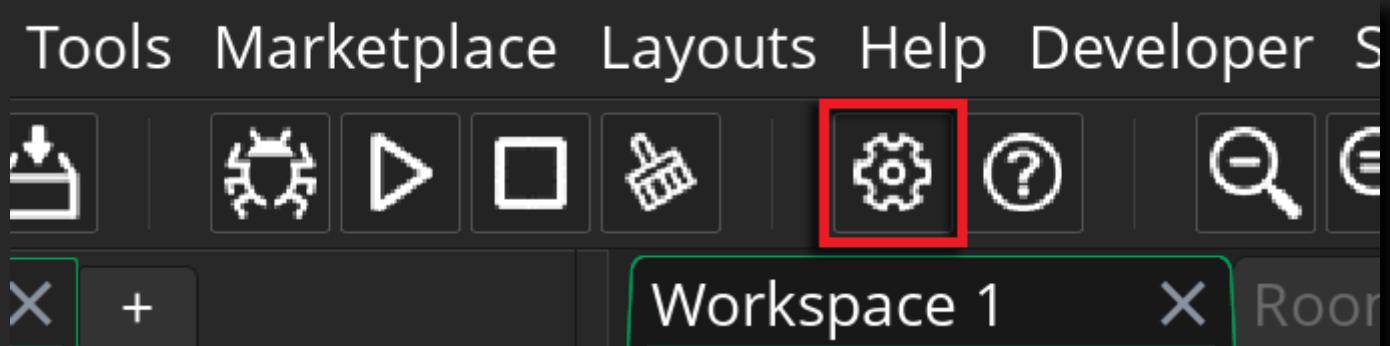
13. Press the Open in Explorer button:



14. Place your **index.html** file inside the folder that opens (/datafiles).

Name	Date modified	Type
WebView	9/14/2021 7:44 AM	File folder
index.html	9/13/2021 9:12 AM	Chrome H

15. Back in GameMaker, click on the **Game Options** button.



16. Go into the **HTML5** platform settings

Game Options - Main

- ▲ Main Options
 - General
- ▲ Platform Settings
 - Opera GX
 - Windows
 - macOS
 - Ubuntu
 - HTML5** ←
 - Android
 - Amazon Fire
 - iOS
 - tvOS

17. In the Advanced section go to the "Include file as index.html" dropdown and select the **index.html** option (this is the file we have just added to the included files).

HTML5 - General

Created with GameMaker Studio 2

Browser Title

1 0 0 0 Version

html5game Folder Name

index.html Output Name

▲ Options

- Output debug to console
- Display cursor
- Display "Running outside server" alert

« ▲ Advanced

Use Default

Included file as index.html

Use Default

index.html

Loading bar extension

18. Press **Apply** and the main setup for all Firebase Web modules is finished!

FirebaseRemoteConfig_FetchAndActivate

This function asynchronously fetches and then activates the fetched configs. If the time elapsed since the last fetch from the Firebase Remote Config backend is more than the set minimum fetch interval, configs are fetched from the backend. After the fetch is complete, the configs are activated so that the fetched key-value pairs take effect.

This is an asynchronous function that will trigger the **Async Social** event when the task is finished.

Syntax:

```
Fi rebaseRemoteConfi g_FetchAndActi vate()
```

Returns:

N/A

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "Fi rebaseRemoteConfi g_FetchAndActi vate"
success	boolean	Whether or not the function task succeeded.

Example:

```
Fi rebaseRemoteConfi g_FetchAndActi vate()
```

In the code above we are fetching and activating the Remote Config data from the Firebase servers. The function callback can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == "FirebaseRemoteConfig_FetchAndActivate")
{
    if (async_load[? "success"])
    {
        var keysJSON = FirebaseRemoteConfig_GetKeys();
        var keys = json_parse(keysJSON);
        var count = array_length(keys);
        show_debug_message("Remote Config Data:");

        for(var i = 0 ; i < count; i++)
        {
            var key = keys[i];
            show_debug_message("key: " + FirebaseRemoteConfig_GetString(key));
        }
    }
}
```

The code above matches the response against the correct event **type**, and if the function task succeeded it gets all keys (using the function **FirebaseRemoteConfig_GetKeys**) that are afterwards parsed into an array (using the **json_parse** function). Finally it gets all their values as strings (using **FirebaseRemoteConfig_GetString**) and logs them to the console.

FirebaseRemoteConfig_GetDouble

This function returns the parameter value for the given key as a real. It evaluates the value of the parameter in the following order:

1. The activated value, if the last successful [FirebaseRemoteConfig_FetchAndActivate](#) contained the key.
2. The default value, if the key was set with [FirebaseRemoteConfig_SetDefaultsAsync](#).
3. The default real value: 0 (zero).

Syntax:

```
FirebaseRemoteConfig_GetDouble(key)
```

Argument	Description
key	The key of the remote config parameter to get the value from.

Returns:

N/A

Example:

```
var magicNumber = FirebaseRemoteConfig_GetDouble("MagicNumber")
```

The code above reads the **double** value parameter for the key "MagicNumber" that is stored in the Remote Config Extension map.

FirebaseRemoteConfig_GetKeys

Returns a JSON formatted string of all the available keys.

Syntax:

```
FirebaseRemoteConfig_GetKeys();
```

Returns:

String (json formatted string)

Example:

```
var keysJSON = FirebaseRemoteConfig_GetKeys();
var keys = json_parse(keysJSON);
var count = array_length(keys);
show_debug_message("Remote Config Data: ");

for(var i = 0 ; i < count; i++)
{
    var key = keys[i];
    show_debug_message("key: " + FirebaseRemoteConfig_GetString(key));
}
```

The code above gets all keys (using the function [FirebaseRemoteConfig_GetKeys](#)) that are afterwards parsed into an array (using the [json_parse](#) function). Finally it gets all their values as strings (using [FirebaseRemoteConfig_GetString](#)) and logs them to the console.

FirebaseRemoteConfig_GetString

This function returns the parameter value for the given key as a string. It evaluates the value of the parameter in the following order

1. The activated value, if the last successful [FirebaseRemoteConfig_FetchAndActivate](#) contained the key.
2. The default value, if the key was set with [FirebaseRemoteConfig_SetDefaultsAsync](#).
3. The default string value: `""` (empty string).

Syntax:

```
Fi rebaseRemoteConfi g_GetDoubl e(key)
```

Argument	Description
key	The key of the remote config parameter to get the value from.

Returns:

```
String
```

Example:

```
var configuration = Fi rebaseRemoteConfi g_GetString("Confi guration")
```

The code above reads the **string** value parameter for the key `"Confi guration"` that is stored in the Remote Config Extension map.

FirebaseRemoteConfig_Initialize

This function initializes the Firebase Remote Config extension and sets the minimum interval between successive fetch calls. Any fetches performed within the interval of the last fetch will use values returned during that fetch.

NOTE The default fetch interval is 3600 seconds.

Syntax:

```
Fi rebaseRemoteConfig_Initialize(fetchInterval)
```

Argument	Type	Description
fetchInterval	real	The minimum fetch Interval duration in seconds.

Returns:

N/A

Example:

```
Fi rebaseRemoteConfig_Initialize(1000)
```

The code above initializes the Firebase Remote Config extension and sets the fetch interval to 1000 seconds.

FirebaseRemoteConfig_Reset

This function deletes all activated, fetched and defaults configs and resets all Firebase Remote Config settings.

This is an asynchronous function that will trigger the **Async Social** event when the task is finished.

Syntax:

```
Fi rebaseRemoteConfig_Reset()
```

Returns:

N/A

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "Fi rebaseRemoteConfig_Reset"
success	boolean	Whether or not the function task succeeded.

Example:

```
Fi rebaseRemoteConfig_Reset();
```

In the code above we request for the reset of the Remote Config value. The function callback can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == "Fi rebaseRemoteConfig_Reset")  
{  
    if (async_load[? "success"])  
    {  
        show_debug_message("FirebaseRemoteConfig_Reset() SUCCEEDED")  
    }  
}
```

```
else
{
    show_debug_message("FirebaseRemoteConfig_Reset() FAILED")
}
```

The code above matches the response against the **correct event type**, and logs the success of the operation.

FirebaseRemoteConfig_SetDefaultsAsync

This function asynchronously sets the default configs using the given JSON formatted string with a set of key-value pairs. Contained values can only be strings or reals.

This is an asynchronous function that will trigger the **Async Social** event when the task is finished.

Syntax:

```
Fi rebaseRemoteConfig_SetDefaultsAsync(jsonString)
```

Argument	Type	Description
jsonString	string	The json formatted representation of a struct with default key/value pairs.

Returns:

N/A

Triggers:

Asynchronous Social Event

Key	Type	Description
type	string	The string "Fi rebaseRemoteConfig_SetDefaultsAsync"
success	boolean	Whether or not the function task succeeded.

Example:

```
var jsonData = { name: "John", age: 10 };

Fi rebaseRemoteConfig_SetDefaultsAsync(json_stringify(jsonData));
```

In the code above we request to set the default values for the parameters inside the struct (`jsonData`) the struct is then converted to JSON formatted string (using the built in function `json_stringify`). The function callback can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == "FilebaseRemoteConfig_SetDefaultAsync")
{
    if (async_load[? "success"])
    {
        show_debug_message("Default values were set successfully!");
    }
    else
    {
        show_debug_message("Default values failed to be set!");
    }
}
```

The code above matches the response against the **correct event type**, and logs the success of the operation.

Platform Setup

Firebase Cloud Messaging implementation uses SDK dependencies and therefore is only available on **Android** and **iOS** targets. In this section we will cover the required setup necessary to start using the Cloud Messaging extension on your game.

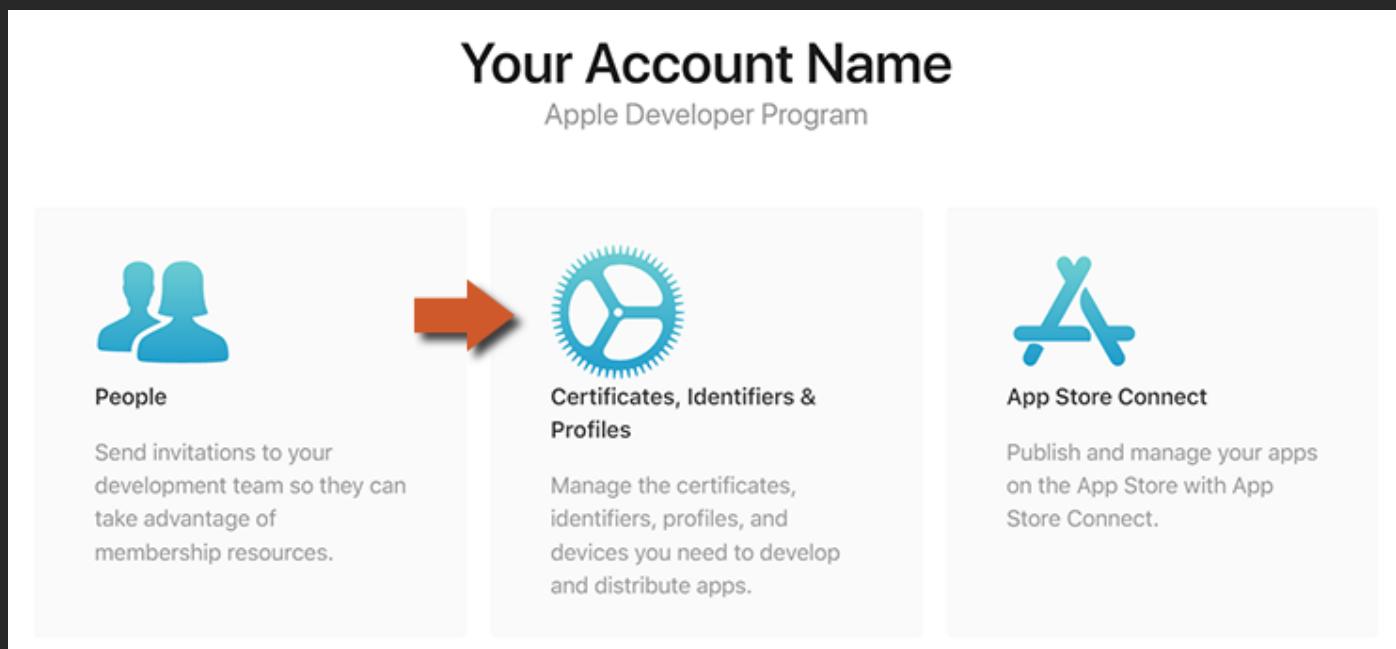
Select your target platform below and follow the simple steps to get your project up and running (you only need follow this setup once per project):

- [Android Setup](#)
- [iOS Setup](#)

Additional steps for iOS

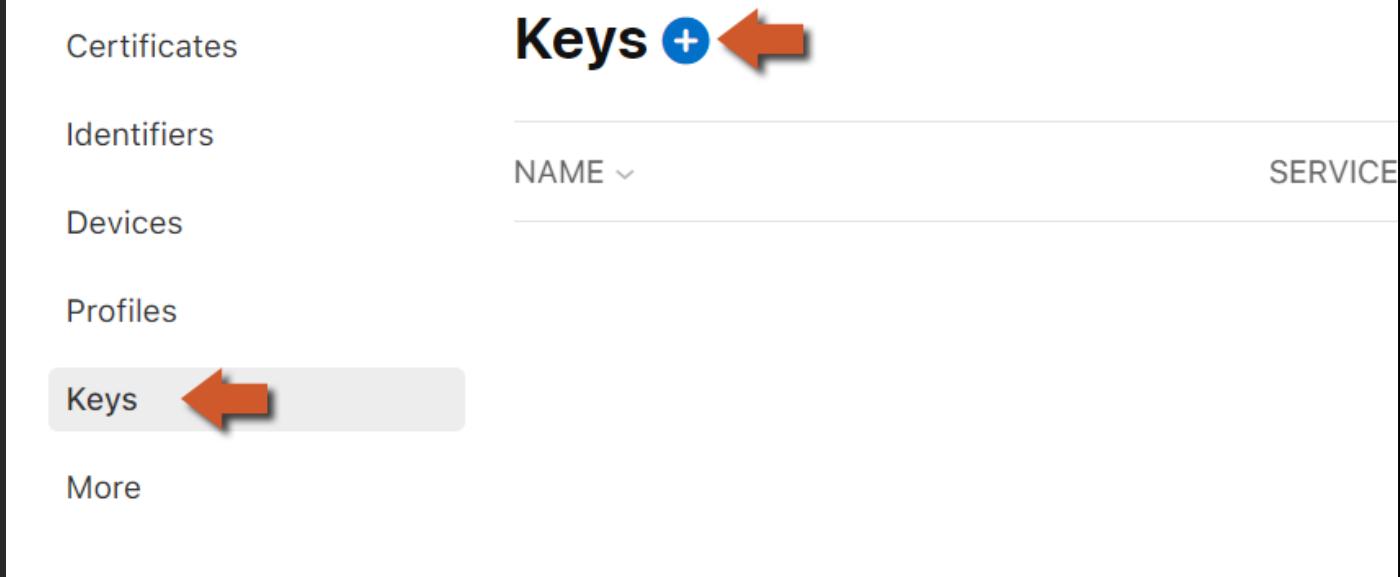
On iOS you will need to retrieve a P8 certificate and upload it to your Firebase project to enable sending push notifications through APNs:

- Head to the [Apple Developer](#) site and select "Certificates, Identifiers & Profiles".



- Select "**Keys**" from the menu on the left, and create a new key by clicking on the plus sign.

Certificates, Identifiers & Profiles



- Enter a **name** for the key, enable **Apple Push Notifications service (APNs)** and click on **Continue**.

A screenshot of the 'Register a New Key' screen. It shows a table with two rows. The first row has columns for 'Key Name' (containing 'Firebase APNs') and 'DESCRIPTION' (containing a placeholder). The second row has columns for 'ENABLE' (with a checked checkbox), 'NAME' (containing 'Apple Push Notifications service (APNs)'), and 'DESCRIPTION' (containing a detailed description). A red arrow points to the 'ENABLE' checkbox in the second row.

Register a New Key		
Key Name	You cannot use special characters such as @, &, *, ', ", -, .	
ENABLE	NAME	DESCRIPTION
<input checked="" type="checkbox"/>	Apple Push Notifications service (APNs)	Establish connectivity between your notification server and the Apple Push Notification service. One key is used for all of your apps. Learn more

- On the next page, confirm the key details and click on **Register**.

A screenshot of the 'Register a New Key' confirmation screen. It shows a table with two rows. The first row has columns for 'Key Name' (containing 'Firebase APNs') and 'DESCRIPTION' (empty). The second row has columns for 'ENABLE' (unchecked), 'NAME' (containing 'Apple Push Notifications service (APNs)'), and 'DESCRIPTION' (containing a detailed description). A red arrow points to the 'REGISTER' button at the top right.

Register a New Key		
Key Name		
ENABLE	NAME	DESCRIPTION
<input type="checkbox"/>	Apple Push Notifications service (APNs)	Establish connectivity between your notification server and the Apple Push Notification service. One key is used for all of your apps.

- Note the information given here (key ID) and download the key as you will not be able to see this screen again.

View Key Details

[Download](#)[Edit](#)

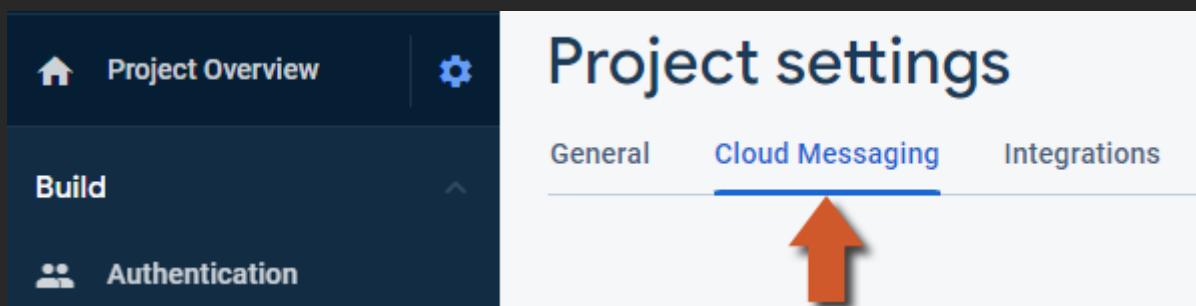
Name
Firebase APNs

Key ID
0000000000

Enabled Services

NAME	CONFIGURATION
Apple Push Notifications service (APNs)	

- Go to the dashboard for your Firebase project and open the **Project Settings**. Here, open the **Cloud Messaging** tab.



- Select your iOS application, and under "APNs Authentication Key", press **Upload** to upload your key.

A screenshot of the 'Apple app configuration' section within the 'Cloud Messaging' settings. It shows a table with two rows. The first row has 'Apple apps' in the header, 'iOS+' status, and 'com.lol.beeglol' listed. The second row contains a note about using APNs authentication keys or certificates. Below this is a section for 'APNs Authentication Key' with a note that auth keys are recommended. It shows a table with columns 'File', 'Key ID', and 'Team ID'. Under 'File', it says 'No APNs auth key'. An orange arrow points downwards from the top of the page towards the 'Upload' button in the 'APNs Authentication Key' section.

- Here, upload your P8 file and enter the other required details that you retrieved from the Apple Developer site.

Upload APNs auth key

X

APNs auth key [?](#)

Drag file here to preview

[Browse](#)

Supports P8

Key ID (required) [?](#)

Enter key ID

Team ID (required) [?](#)

Enter team ID

You can now send notifications to the iOS client game by going under "**Engage**" and selecting "**Cloud Messaging**" on your Firebase dashboard.

