MPLAY Documentation

Last Updated: 19-09-22

1. Introduction

Welcome to **MPLAY** app installation doc & thank you for buying our product.

2. Flutter Installation

So the app is developed on Flutter. So, You have to install flutter on your computer. You can use Both **Visual Studio Code** & **Android Studio** as an IDE for Flutter. The steps are the same for both IDE. We have used Visual Studio Code. So, Our Setup will be based on this IDE.

To install flutter on your computer, follow the official documentation from Google.

Flutter Official Site : https://flutter.dev/.

You can follow these youtube videos to install flutter also.

1. For Mac: https://www.youtube.com/watch?v=9GuzMsZQUYs
2. For Windows: https://www.youtube.com/watch?v=T9LdScRVhv8

Make sure you have installed the latest stable version of flutter. If everything is okay then you can follow the further steps below.

Note: *A Mac (Apple Desktop/laptop) device and an apple developer account is required for the iOS setup. If you don't have both of them, you can ignore the iOS steps.*

3. App Setup

After purchasing the app you will get a .zip file and unzip it. Inside the main folder you will get a folder named **MPLAY.** This is the source code of the app. You have to work on this to set up your app. Now open your IDE and open the **MPLAY** folder on your IDE. Wait some time to load the project.

Now go to the IDE terminal and run the following commands one after one :

```
flutter clean
```

After that, run

```
flutter pub get
```

Wait some time to get all the packages.

4. Change Package Name for Android

Now, you have to change the package name in the source code. You have to use the same package name that you have registered in the firebase console.

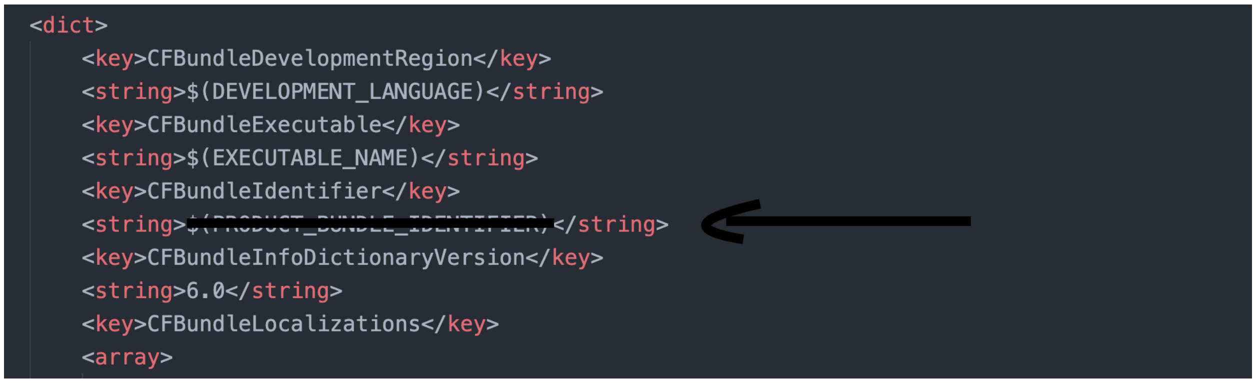
1. Go to your IDE and now you **have to change the package name** of your app. Run the following command from your IDE terminal by changing the package name with yours.

```
flutter pub run change_app_package_name:main com.new.package.name
```

Don't forget to replace **com.new.package.name** with your package name. That's it. Your package name has been changed now.

5. Change Package Name for iOS

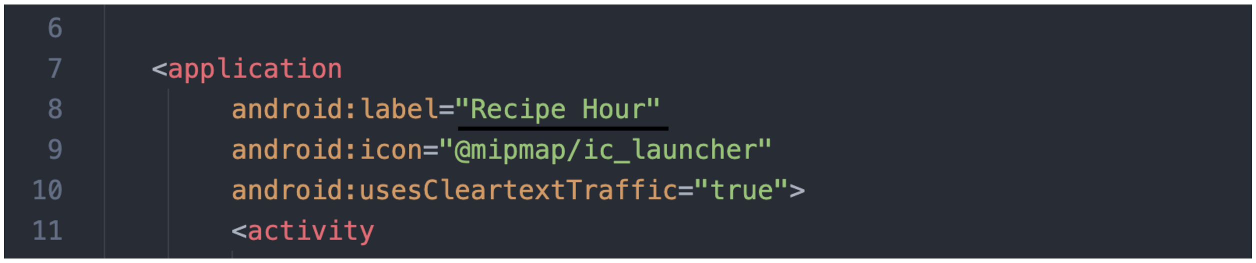
To do that, go to **ios/Runner/Info.plist** file and replace **CFBundleIdentifier** value with your iOS package name. (See the picture below)



```
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>$(DEVELOPMENT_LANGUAGE)</string>
    <key>CFBundleExecutable</key>
    <string>$(EXECUTABLE_NAME)</string>
    <key>CFBundleIdentifier</key>
    <string>~~$(PRODUCT_BUNDLE_IDENTIFIER)~~</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleLocalizations</key>
    <array>
```

7. Change App Name for Android

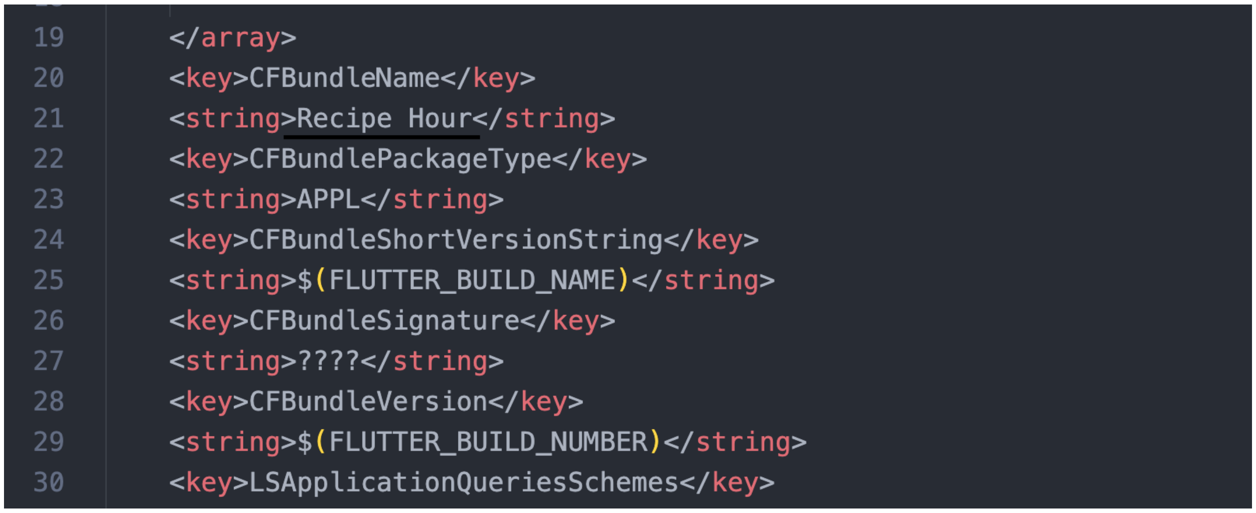
Go to **android/app/src/main/AndroidManifest.xml** file and Change your app name.



```
    <application
        android:label="Recipe Hour"
        android:icon="@mipmap/ic_launcher"
        android:usesCleartextTraffic="true">
        <activity
```

8. Change App name for iOS

Go to **ios/Runner/Info.plist file** and Change your app name.



```
	</array>
	<key>CFBundleName</key>
	<string>Recipe Hour</string>
	<key>CFBundlePackageType</key>
	<string>APPL</string>
	<key>CFBundleShortVersionString</key>
	<string>$(FLUTTER_BUILD_NAME)</string>
	<key>CFBundleSignature</key>
	<string>????</string>
	<key>CFBundleVersion</key>
	<string>$(FLUTTER_BUILD_NUMBER)</string>
	<key>LSApplicationQueriesSchemes</key>
```

9. Change App Icon

See How to generate an application icon?

- Browse your image and click on Download icon. After successfully generated, replace all icons in respective folders:
- /mipmap-hdpi in /android/app/src/main/res/ folder
- /mipmap-mdpi in /android/app/src/main/res/ folder
- /mipmap-xhdpi in /android/app/src/main/res/ folderr
- /mipmap-xxhdpi in /android/app/src/main/res/ folder
- /mipmap-xxxhdpi in /android/app/src/main/res/ folder

10. Android Release Key Setup

To generate a release certificate, You have to generate a keystore file. To generate a keystore file, run this command below from the root of your project directory on the terminal.

For Mac users, run

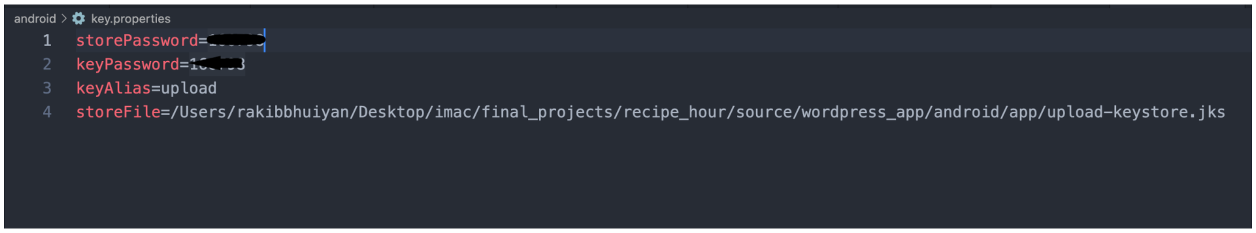
```
keytool -genkey -v -keystore ~/upload-keystore.jks -keyalg RSA -keysize 2048 -validity
10000 -alias upload
```

For Windows users, run

```
keytool -genkey -v -keystore c:\Users\USER_NAME\upload-keystore.jks -storetype JKS
-keyalg RSA -keysize 2048 -validity 10000 -alias upload
```

1. Enter your details and remember the **password**. After this, you will get an **upload-keystore.jks** keystore file.

2. Locate this file and move the file into the **android/app** folder and copy the path by right clicking on the **upload-keystore.jks** file
3. Then go to **android/key.properties** file and replace the path of the keystore file of yours. Then also replace the **password** which you have inputted to generate the keystore file.



android > key.properties

```
storePassword=[redacted]
keyPassword=[redacted]
keyAlias=upload
storeFile=/Users/rakibbhuiyan/Desktop/imac/final_projects/recipe_hour/source/wordpress_app/android/app/upload-keystore.jks
```

That's it. Android & iOS App Setup is complete.

11. Run The App

So Your Setup is 100% complete now. Now run this following command to clean the project.

```
flutter clean
```

After that, run this command,

```
flutter run
```

And After that run the following command to run this app on your physical or emulator devices. Make sure you have connected an android/ios emulator or connected an real device via USB.

```
flutter run
```

Test if everything is okay or not.

12. Release The Android App on Google Play Store:

You have done all the things that are required for android release. To Test the release android app, run the following command on the terminal.

```
flutter build apk --split-per-abi
```

You will get 3 apk files from the **build/app/output/apk/release** folder. You can test the **v7** version of the apk file. If you want to publish the app in the google play store, don't upload any of the following files. Use an **appbundle** file which is recommended by Google. To generate an appbundle, run the following command on terminal :

```
flutter build appbundle
```

After that, you will get an **.aab** file in the **build/app/output/appbundle/release** folder. Now you can upload this .aab file to the google play store.

13. Release the iOS app on App Store :

Follow the official doc from flutter team here.

14. Conclusion

That’s it. We know that you are so tired right now. Take some rest. Everything is complete now.