# Agora-Unity

**Install Agora SDK v3.7.0.3.**

Here, I will explain you about how to use Video Call in Unity using agora and taking the data from the data bucket.

**Prerequisites**

● Unity Editor (2017 LTS or above)

● A developer account with Agora.io

**Add Your AppID**

Before you can build and run the project, you will need to add your AppID to the configuration.

Go to your developer account’s project console, create a new AppId or copy the Appd from an

existing project. Perform the following steps:

1. Open the Assets/QuickTest scene,

2. Open Test/QuickTest/AgoraInfo addressables

You have to add your API key in the AgoraInfo addressables which we get previously.

At this step, you should be able to run the QuickTest scene within the Unity Editor. Input your desired channel name to the input field and click Join. You will be able to connect to the channel which has the same appID and channelID.

**Explaining Custom Features**

You can also customise the data like InstructorID, ChannelName ,Video height , width , Bitrate(through which the data will be processed),Audio Indication intervals, FrameRate of the video, Orientation , degradation prefrance, Video Quality, Audio Profile, Audio Scenario, ChannelProfile and you can even add some new ideas.

InstructorID :

Here you can assign the InstructorID with which you want to connect with the instructor.This was basically designed as I don’t want to assign the same role to instructor and the student

ChannelName :

Here you can assign the channel name with whom you want the user to be connected.

Video height and width:

As I want myself to have full control over my video during the video call. I can adjusted the video height and width.

Bitrate :

As I want to assign the data rate to not give load on the server. So, here I can adjust the data rate

Audio Indication intervals:

FrameRate of the video:

Orientation :

Degradation prefrance:

Video Quality:

Audio Profile:

Audio Scenario:

ChannelProfile:

**Player Settings for Building the Sample Application**

Common Setting

Open the Build Settings and drag SceneHome and SceneHelloVideo scenes from the assets list into the “Scenes in Build” list.

Setting Plugin Identities

Normally the library identities have been setup with the bundled SDK plugins. In case of manual override needed, follow the following steps:

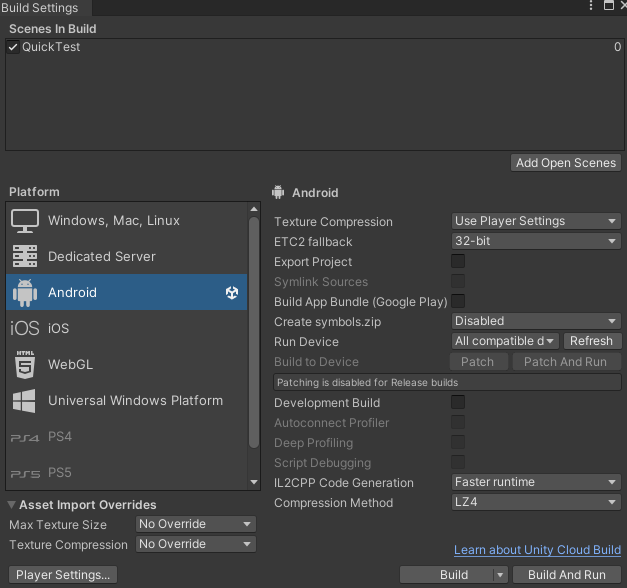
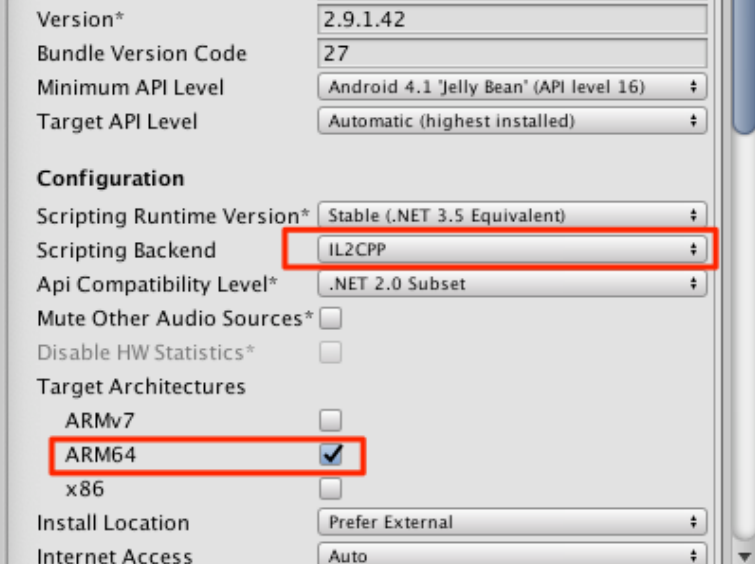
For 64-bit Windows builds, go into Assets->Plugins->x86\_64 folder, select “Editor” and “Standalone” and then click Apply.

To ensure that the build plugin libraries don’t collide, disable the identities of the files in the x86 folder.

Do the opposite for 32-bit Windows.

For MacOS, make sure Any CPU is selected. It is not automatically done for Unity version 2020.2 and up especially. See details in the MacOS Build section below.

Android Build

Select Android from the platform list and click Switch Platform.  

Once Unity finishes the setup process, open the Player Settings and set a unique package name,

In order to comply with Google's 64 bit App requirement, make sure the following setting is selected:

1. Change the Scripting backend to IL2CPP.

2. Select ARM64 for the Target Architecture

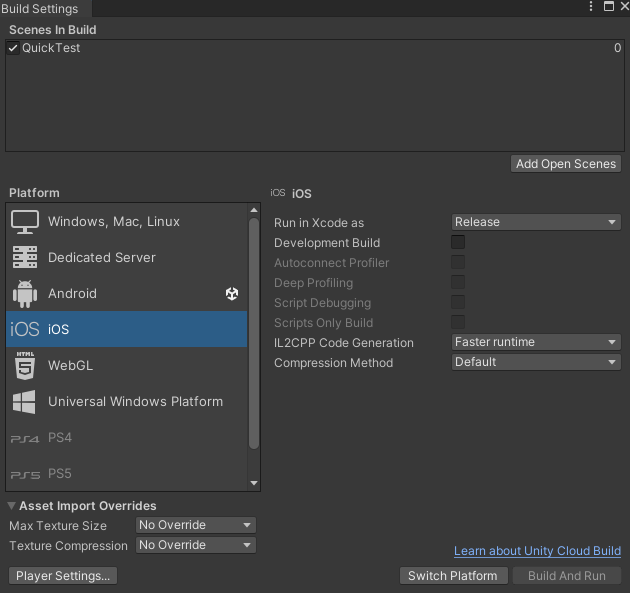
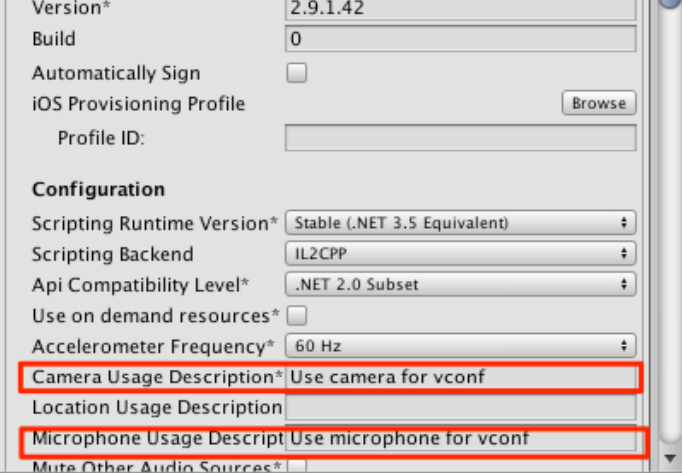
Leave other settings as is. For AR/VR enabled Applications, please refer to the separate README file.

iOS Build

Select iOS from the platform list and click Switch Platform.

The default build setting should work for most cases. The only custom settings are:

1. Change the Bundle Identifier to your own Bundle identifier so XCode can properly codesign the application.
2. Ensure the microphone permission has a description to allow the user to know why the microphone is being accessed by the application
3. Ensure the camera permission has a description to allow the user to know why the camera is being accessed by the application

**Embedding Frameworks**

Starting with SDK version 3.0.1, the iOS plugin frameworks are dynamically loaded and need to be embedded into the libraries. The Post Processing build script BL\_BuildPostProcess.cs should have taken care of this. If your iOS build runs perfectly, then no need to read on for the following help text.

Just in case of anything missed on building your own projects, please be sure that all the iOS Frameworks and library are put into Embedded section for XCode:

Make sure you do for all the items in the iOS folder:

Windows Build

Select “PC, Mac & Linux Standalone” from the platform list and click Switch Platform. Then Select “Windows” for the Target Platform. Also choose x86 for 32 bit or x86\_64 for 64 bit architecture according to your Application target. Windows Build Remember to set the appropriate Plugin library identities as described in the earlier section of this README file.

