

# Agora Video SDK README

This document shows how you can configure and run your Unity Application with Agora IO's Video SDK. The SDK supports iOS, Android, MacOS and Windows platform on Unity.

## Prerequisites

- Unity Editor (2017 LTS or above)
- A [developer account](#) with Agora.io

## Getting Started

Although you may start the integration with your existing project, in this short tutorial we will just use the demo project included in the SDK.

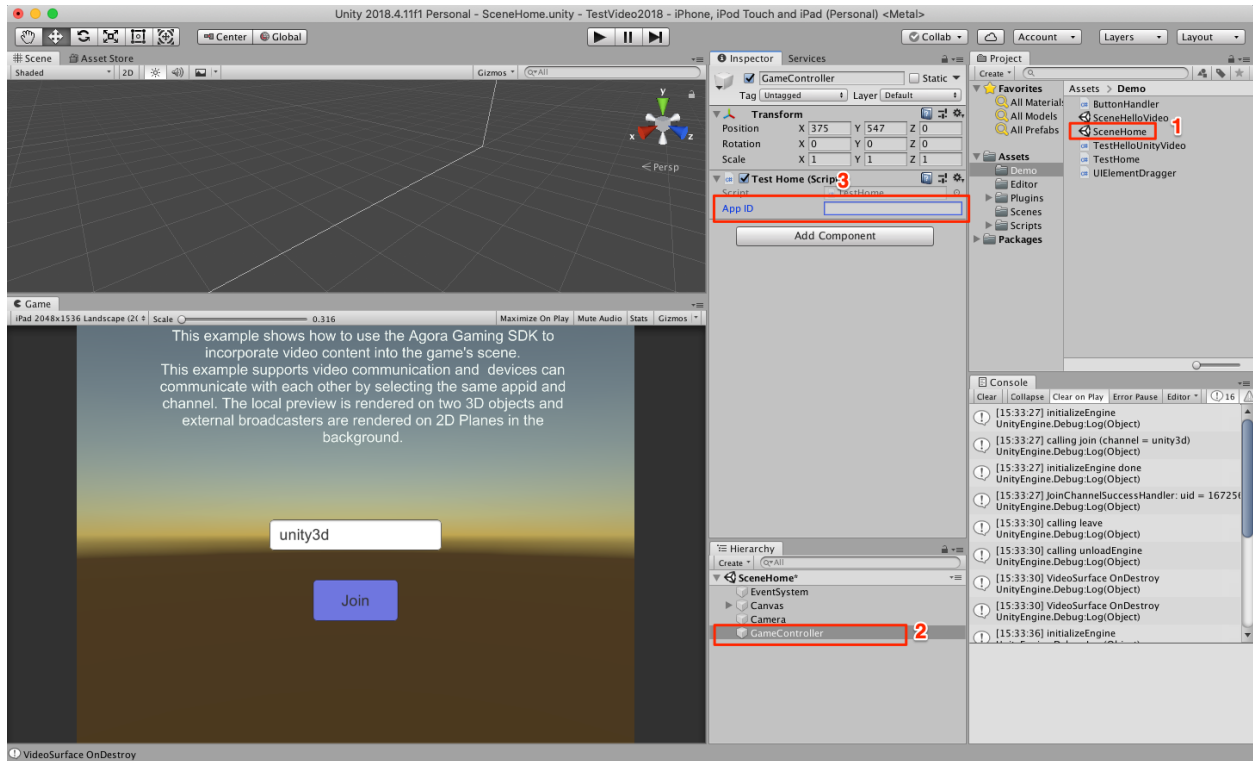
Open a new Unity project and navigate to Unity Asset Store and search for "Agora Video SDK". Download and import the assets. Please scroll to the last section about upgrading if you are replacing the SDK on top of the old one.

## Add Your AppID

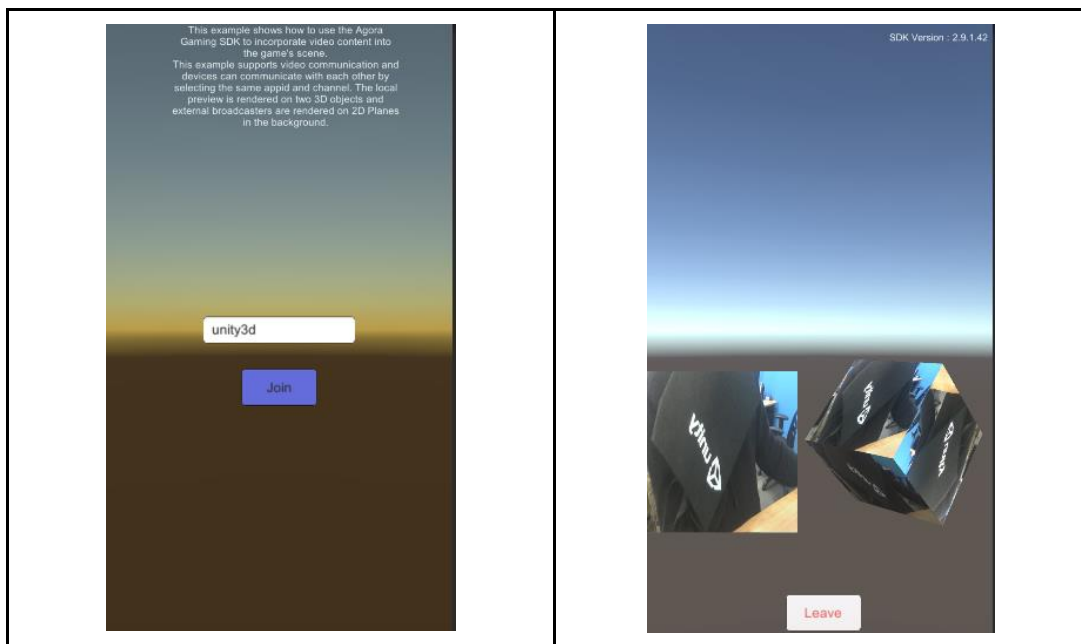
Before you can build and run the project, you will need to add your AppID to the configuration. Go to your [developer account's project console](#), create a new AppId or copy the Appd from an existing project. Perform the following steps:

1. Open the Assets/Demo/SceneHome scene,
2. Select the GameController from the Unity Editor's Hierarchy panel.
3. The GameController game object has a property App ID, this is where you will add your Agora App ID.

See the following screen capture:



At this step, you should be able to test the demo App within the Unity Editor. Input your desired channel name to the input field and click Join. You will be taken to the next scene and see a Quad and a Cube showing the local camera stream.



# 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.

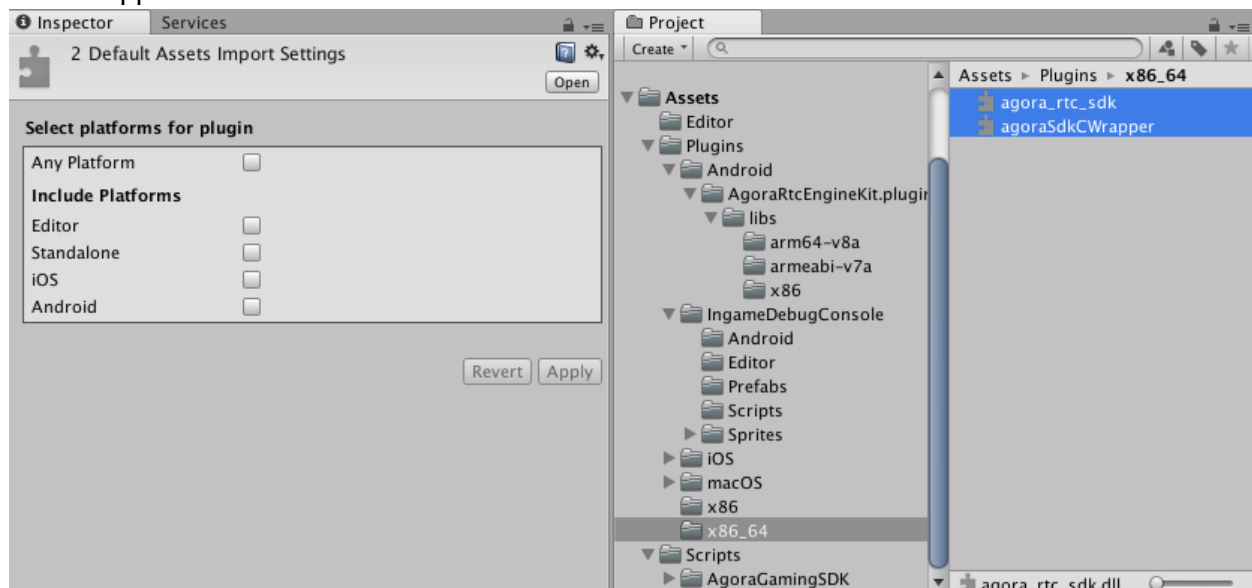
## Setting Plugin Identities

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.

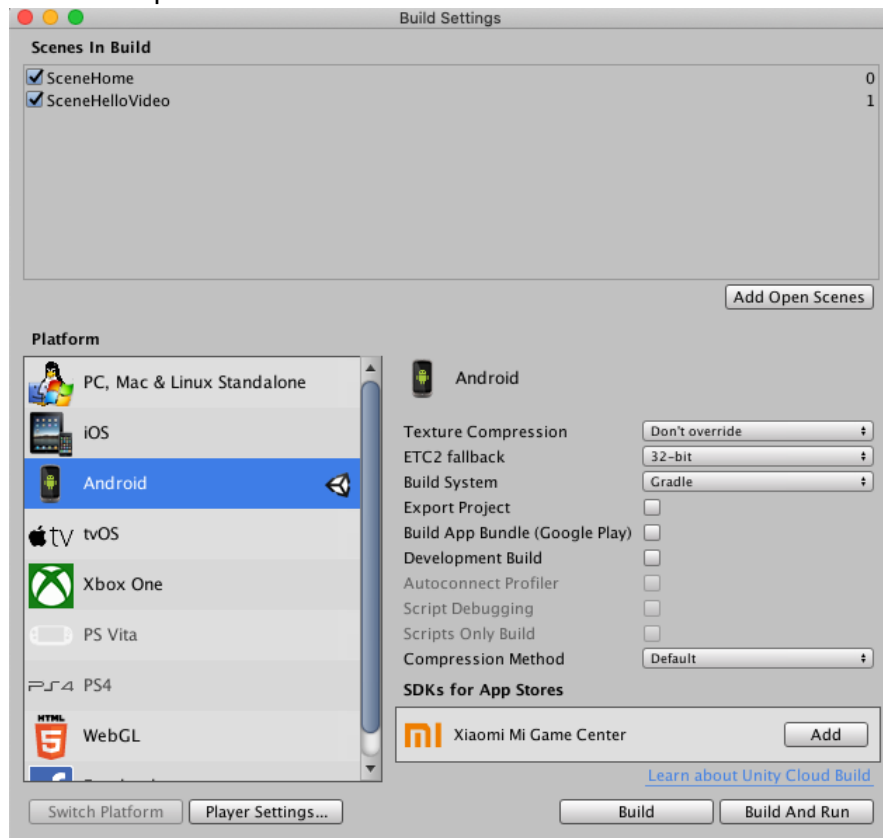
Do the opposite for 32-bit Windows.



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.

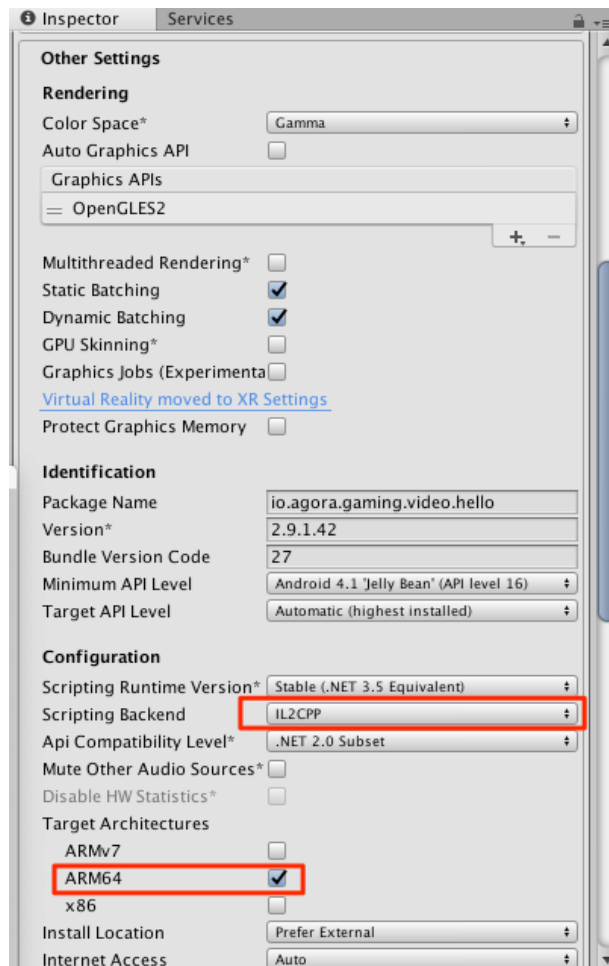


### Android Build

Once Unity finishes the setup process, open the Player Settings and set a unique package name, e.g., io.agora.gaming.video.hello.

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.



Android Build Settings

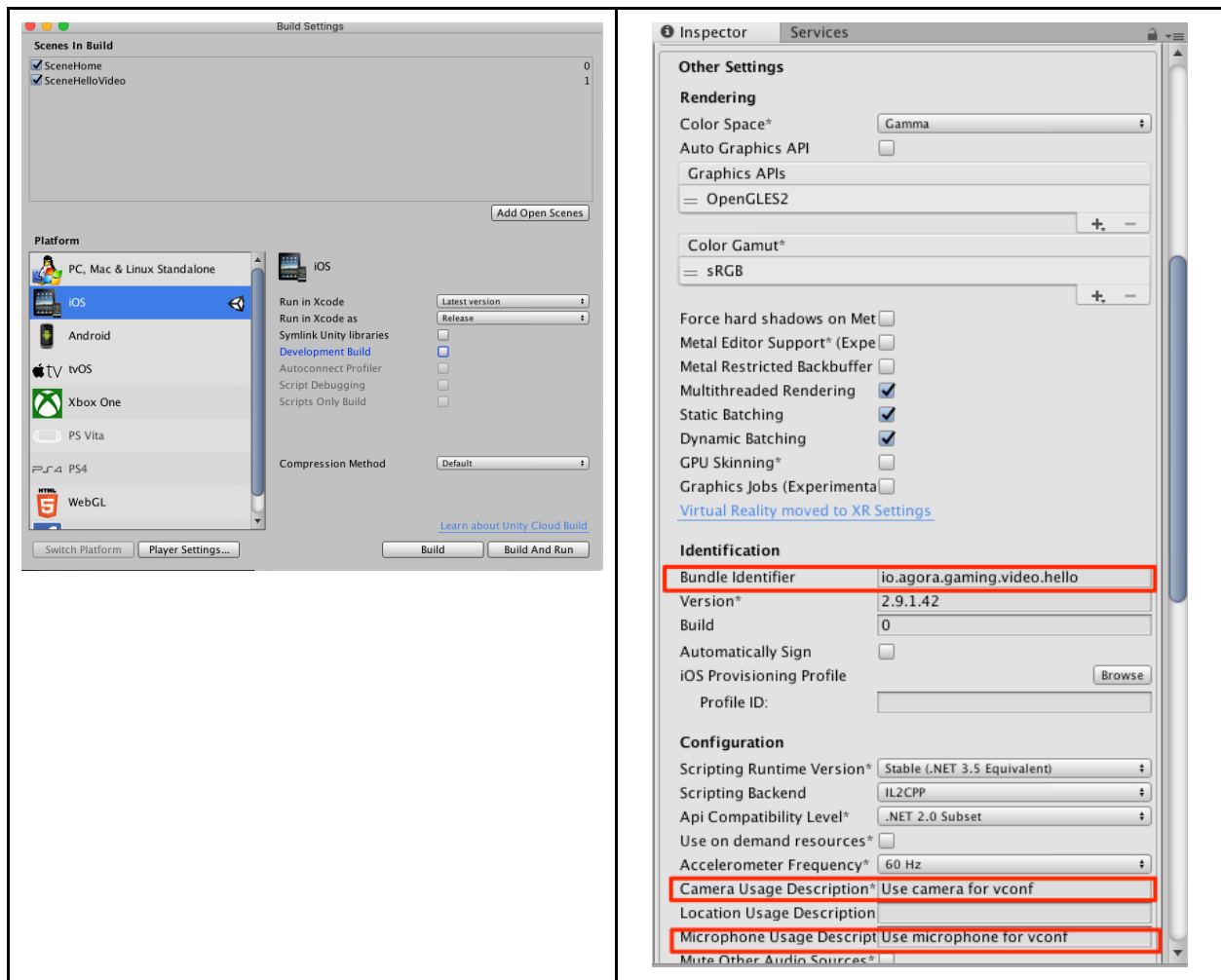
Leave other settings as is. For AR/VR enabled Applications, please refer to the separate README file.

## 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



iOS Build Settings

## Embedding Frameworks

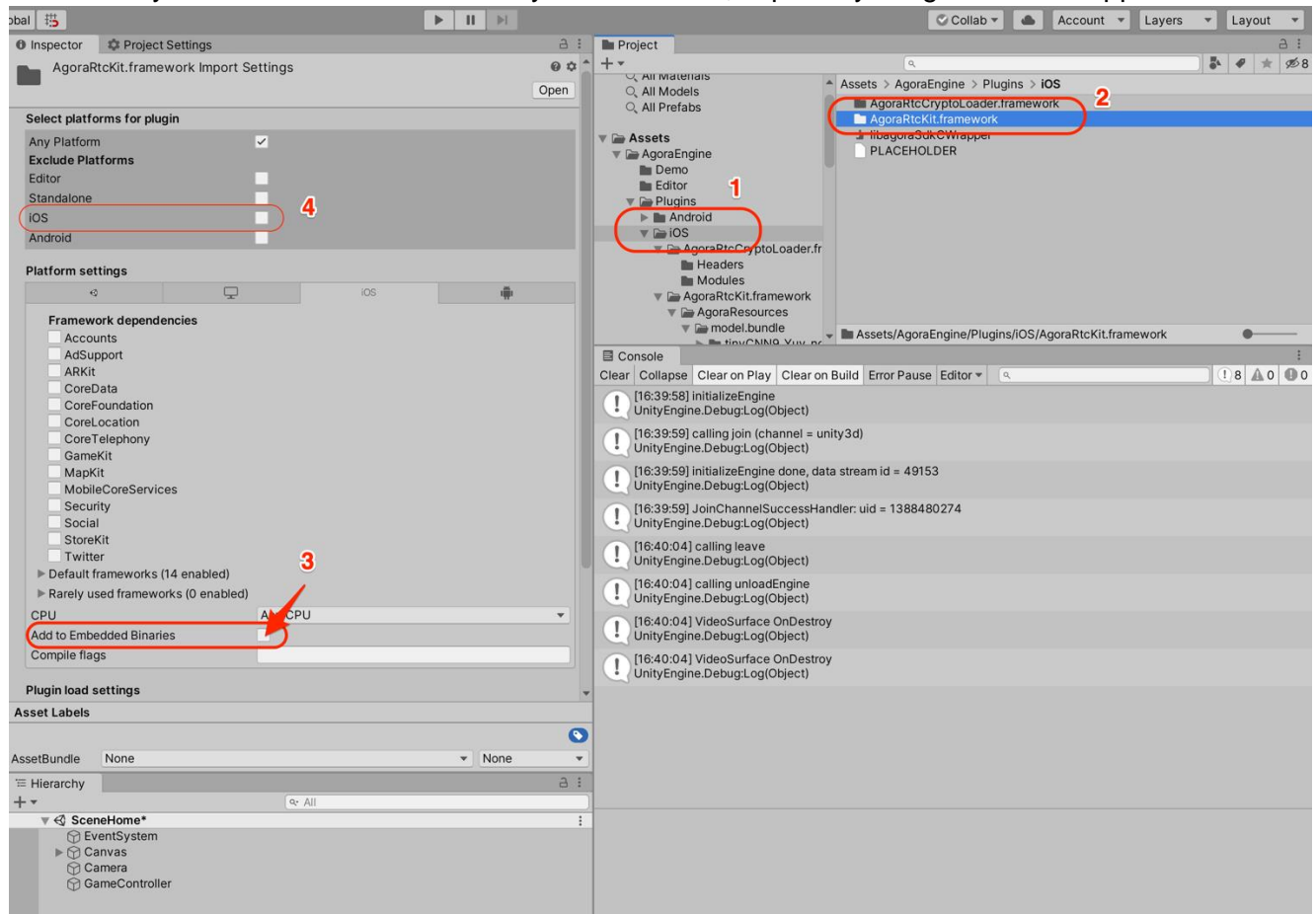
Starting with SDK version 3.0.1, the iOS plugin frameworks are dynamically loaded and need to be embedded into the libraries. The Post Processing build script BL\_BuildPostProcess.cs should have taken care of this. If your iOS build runs perfectly, then no need to read on for the following help text.

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

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

- AgoraRtcCryptoLoader.framework
- AgoraRtcKit.framework
- libagoraSdkCWrapper

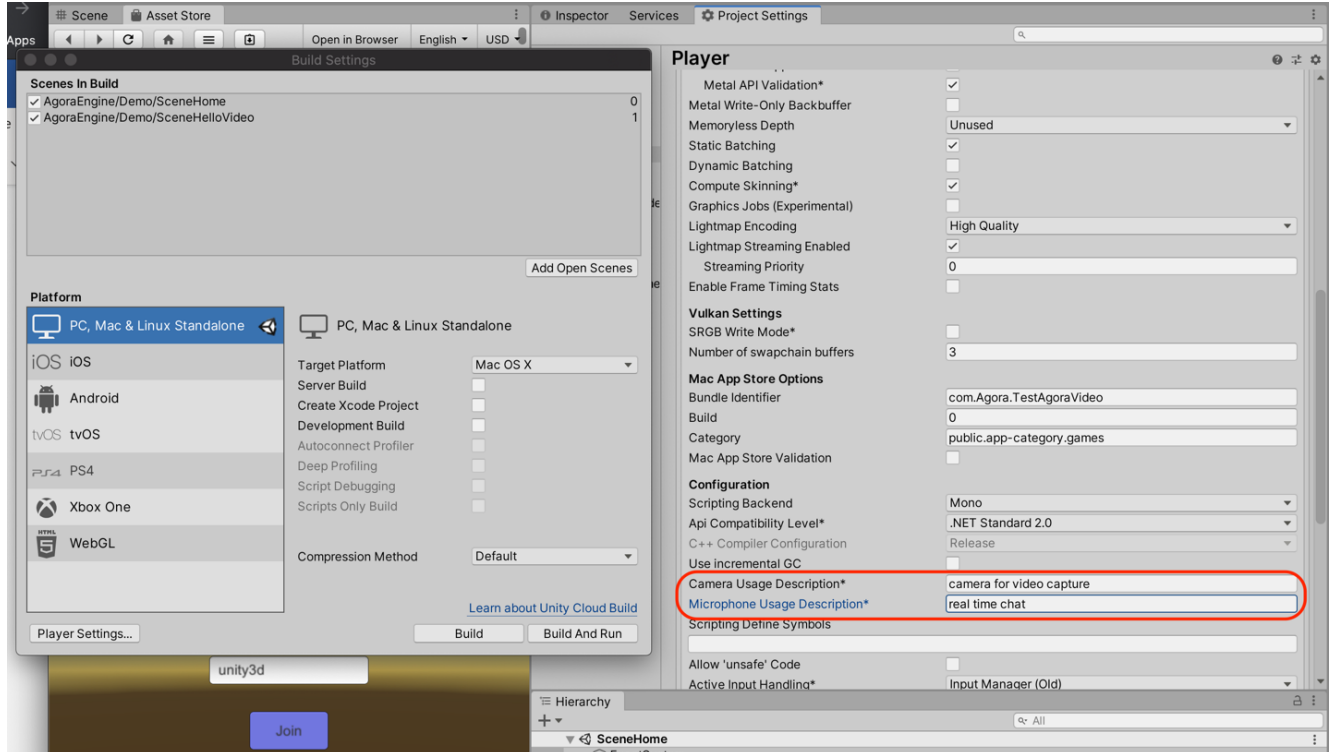
Make sure you choose **iOS** as the identity for these files, especially libagoraSdkCWrapper.



Unity Editor Embedded Binary setting (Unity 2018,2019 and up)

## MacOS Build

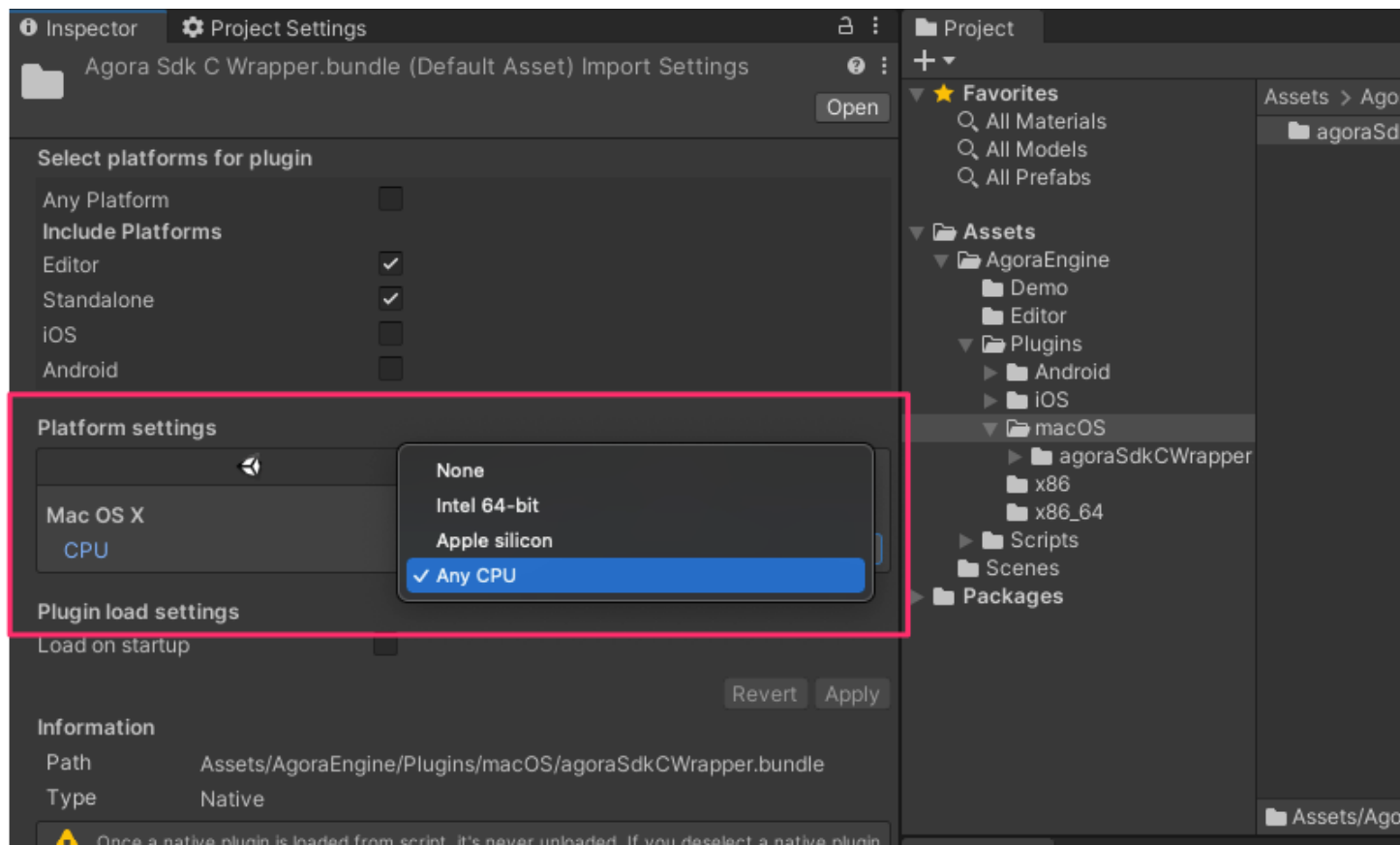
Select “**PC, Mac & Linux Standalone**” from the platform list and click Switch Platform. Then Select “**Mac OS X**” for the Target Platform. Besides regular App Store required information, it is **very important** to fill in the **Camera and Microphone** usage description. Without this your app may **crash** on MacOS Catalina or greater for privacy restriction reasons.

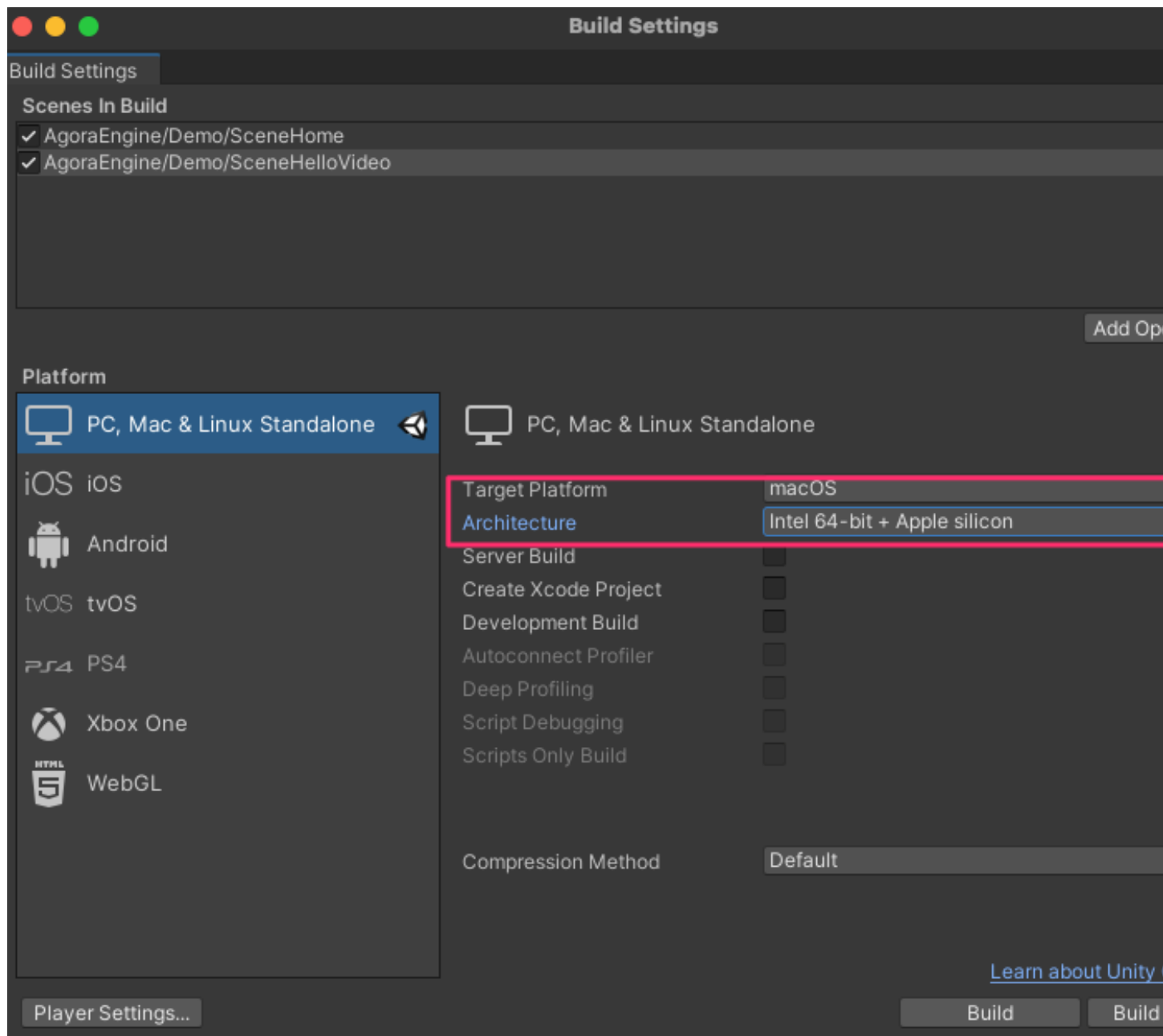


MacOS Build Settings

Starting from Unity Editor 2020.2 and MacOS Big Sur, Apple made M1 CPU is added in the Architecture options. Make sure you configure **agoraSdkCWrapper.bundle** module to match the target in Build Settings. See follow screenshots:







## Codesigning for Notarization

For Notarization validation, Agora library frameworks need to be code-signed with the main build executable together. The SDK provided helper scripts to get your signing step more convenient. See instructions below.

## Codesign instruction after mac build

use scripts provided in Unity SDK project folder: **Assets/AgoraEngine/Scripts/AgoraTools**  
Assume the project folder is `${project_path}`, your build is output to `${build_folder}`, your app is `TestMAC.app`

### Steps:

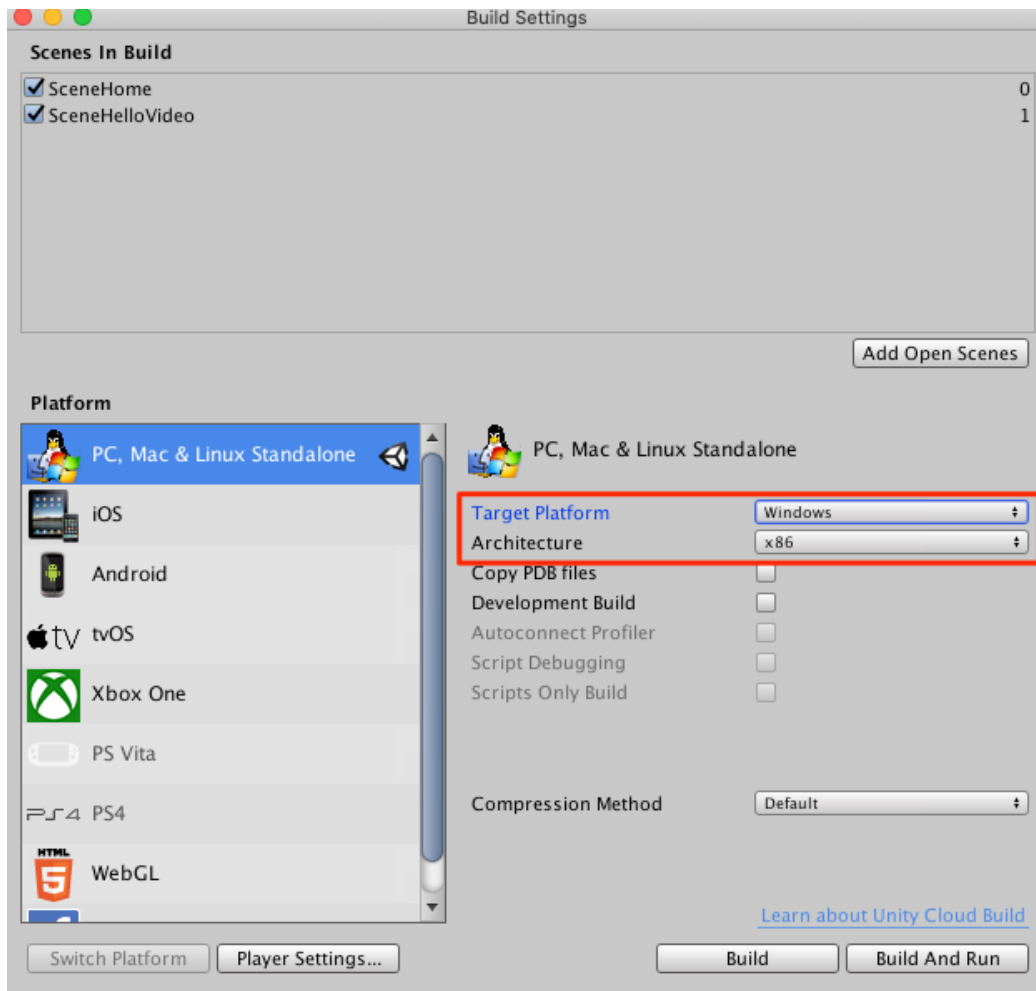
1. `cd ${build_folder}`
2. `${project_path}/Assets/AgoraEngine/Scripts/AgoraTools/prep_codesign.sh`  
`TestMAC.app`
3. find the signature on the Mac with this command `/usr/bin/security find-identity -v -p codesigning` (hint from the script too)
4. `(export SIGNATURE="<your signature>";`  
`${project_path}/Assets/AgoraEngine/Scripts/AgoraTools/signcode.sh TestMAC.app)`
5. Verify that the result is

**TestMac.app: valid on disk**

**TestMac.app: satisfies its Designated Requirement**

## 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.

# Upgrading SDK

From versions prior to 2.9.1:

- Remove files according to this SDK directory tree: <https://bit.ly/3eO16Jy>
- Clean up Unity Asset cache (explained later)

From version 2.9.x to 3.x.x:

- Remove the entire file tree under Assets/AgoraEngine
- Importing the new SDK

## How to **Clean up old Unity Asset Cache**

- Refer to this documentation link to find the location of the cache folder:  
<https://docs.unity3d.com/Manual/AssetPackages.html>
- Note that the Asset Store fold may have a version attached to it. For example, this is the current location of the cache to the Agora Video SDK on MacOS:

~/Library/Unity/Asset Store-5.x/Agoraio/ScriptingVideo

## Testing

After the configuration has been set up according to the above rules, you should be able to build to the target platform. The following is the sample demo running on device. The remote player will show up in the RawImage generated by code in the upper screen area. User may drag the image around to change its position when more players join the video chat channel.



## Some Trouble Shooting FAQs

**Q:** There are only white cube and box on the demo screen after I join the channel.

**A:** There may be definite reasons for this. First recommendation is implementing the OnError and OnWarning callbacks to check on what errors are occurring. (Added in the Demo starting from v3.2.1.71). Here are the common causes:

1. You registered an AppId with Certificate enabled, but you didn't include a secure token in your JoinChannel call. => You should use an AppId without Certificate if you are new to the SDK. Create a new AppId with Certificate after you tested the POC and feel comfortable to enforce token security.

2. The Application did not acquire the required OS level permission for Camera. => Make sure you set those up in the PlayerSettings and Manifest if for Android.
3. The agoraCWrapper library didn't get loaded on run-time (usually happens on Standalone, not Editor). Make sure you have the correct Platform Settings flag selected. See examples in MacOS section for building the App above.

**Q:** Do you support WebGL?

**A:** No, not in this version. But it is on the roadmap.

**Q:** App crashed! Why?

**A:** 99% is that the privacy policy is violated on the running OS. Take MacOS for example, you must declare the description of the camera usage in the Player Settings. Sometimes, your standalone App couldn't get the permission even you had set it in the Unity Player Settings. You must find ways to make sure that is taken care of at the OS setting level. There are solutions on the Internet you can find. You may also come to the Slack groups to look for help (see the links in Resources below).

**Q:** Create engine gets error with return code -7, why?

**A:** It is likely that your previous run of the code did not clean up properly when program terminates. Make sure you unload the engine by calling `IRTCEngine.Destroy()` inside `OnApplicationQuit()`. Exit Unity Editor and run again.

More FAQs: <https://docs.agora.io/en/Video/faq?platform=Unity>

## Other Resources

- GitHub demos: <https://github.com/AgoraIO/Agora-Unity-Quickstart>
- The complete API documentation is available in the [Document Center](#).
- For technical support, submit a ticket using the [Agora Dashboard](#) or
- Join our slack community: <https://bit.ly/39j4l5h>
- Help each other at <https://agoraiodev.slack.com/messages/unity-help-me>
- Developer relations team: [devrel@agora.io](mailto:devrel@agora.io)
- Release note:  
[https://docs.agora.io/en/Interactive%20Broadcast/release\\_unity\\_video?platform=Unity](https://docs.agora.io/en/Interactive%20Broadcast/release_unity_video?platform=Unity)