Agora Video SDK README

The Agora Unity Video SDK supports iOS, Android, MacOS and Windows platform. For WebGL support, please ask our staff for the Community beta before official support is available. This document helps you quickly get started with the SDK and API-Examples. And it includes resources that will help you on the migration and problem shooting need.

Prerequisites

- Unity Editor (2018 LTS or above)
- A developer account with Agora.io

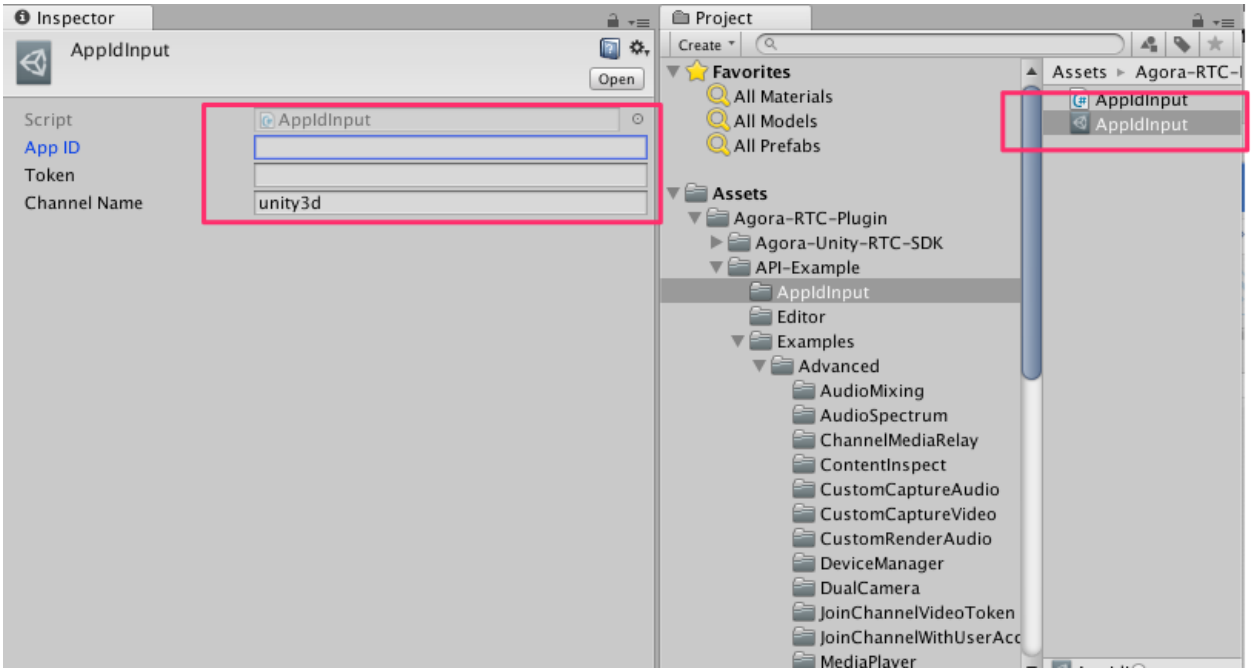
SDK Package Structure

The RTC Plugin contains two main folders:

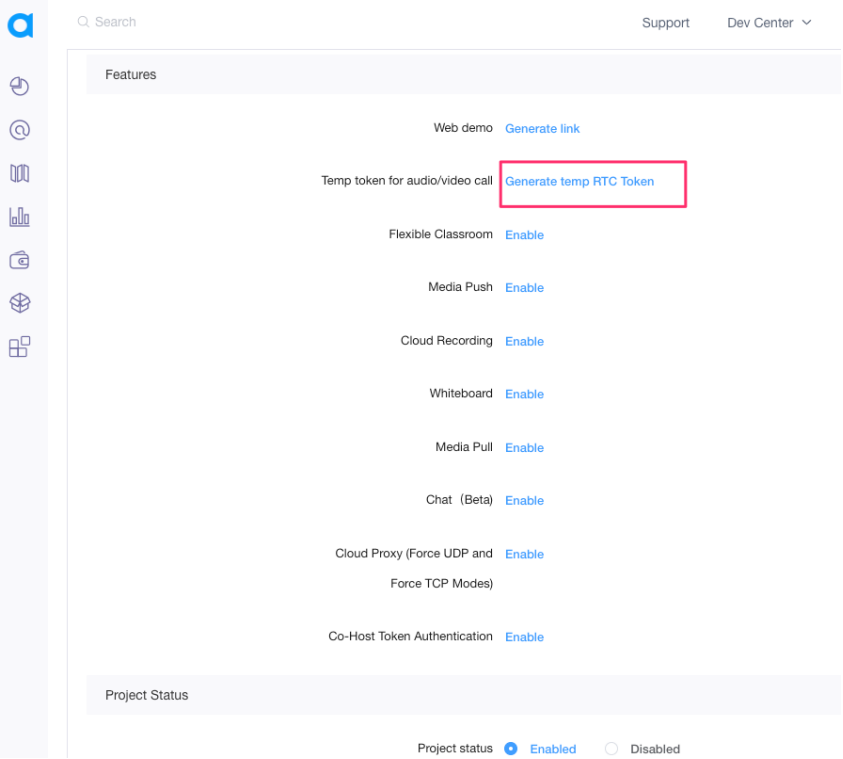
- Agora-Unity-RTC-SDK: contains library frameworks and C# level code.
- API-Example: simple demos that focuses on individual feature APIs. The latest update can be found at the Agora-Unity-Quickstart repo.

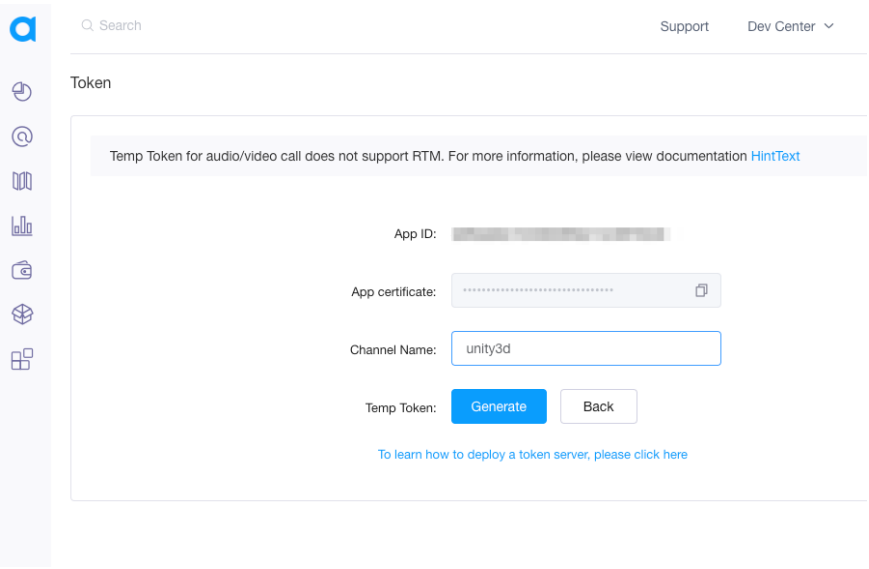
Your AppID

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 AppID from an existing project. Fill that AppID into the **AppIdInput** scriptable asset here:



Note that you will need get a token if you AppID was generated with the Token authentication option. In such case, if you don't have a token server setup already, you may obtain a temporary token from your developer console. Go to **Project Management >More >Config** on that project. Under the **Edit Project** menu, select **Generate Temp RTC Token**. Enter the channel name that you want to use.





If you created your AppID in test mode, then you don't need to do the token generation for this test. But keep in mind that in a production environment, you should use token authentication for better security protection.

At this step, you should be able to test the demo App within the Unity Editor. Select any API-example scene and run the test there.

From SDK version 4.0.0 and newer, a **Home** scene is added at the **API-Examples** folder. You may test all the individual API examples in one build. To do that, just add all the scenes from the subfolders into the build settings.

Player Settings for Building the Sample Application

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 building on a 32-bit Windows machine.

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.

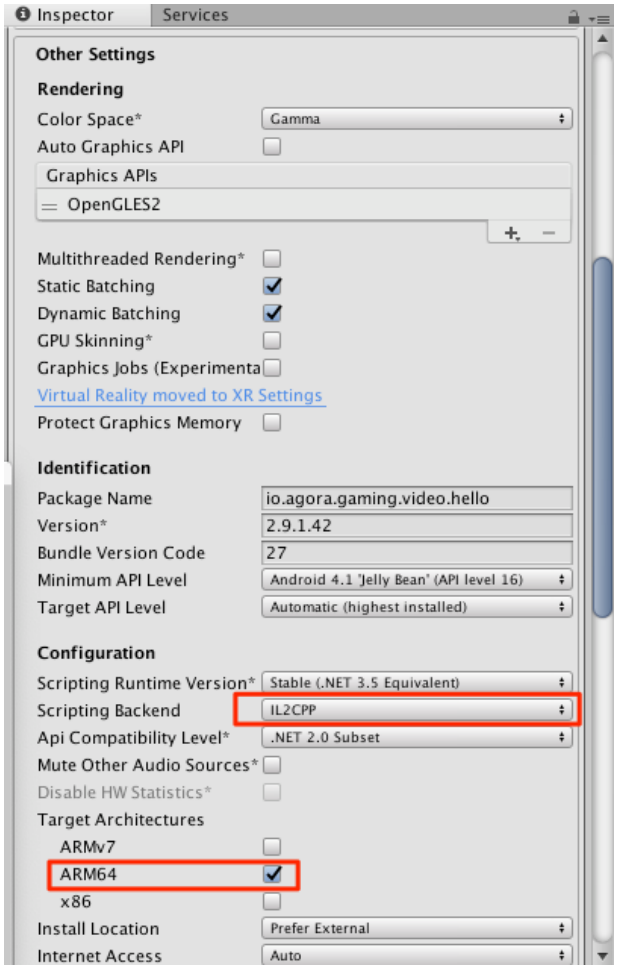
Android Build

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, e.g., *io.agora.gaming.video.hello.*

Google's 64 bit App requirement implies that following settings are desired:

1. Change the Scripting backend to **IL2CPP**.
2. Select **ARM64** for the Target Architecture.



Android Build Settings

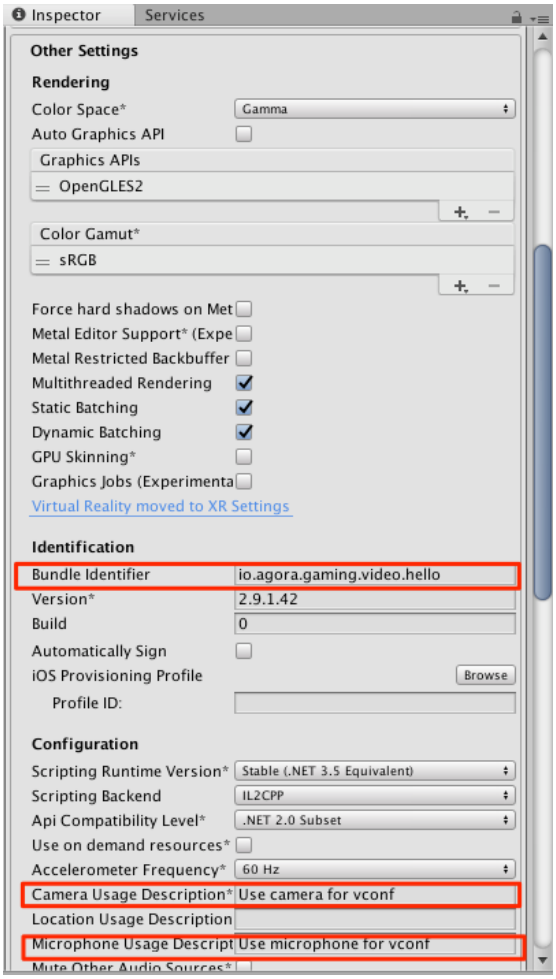
In some versions ranges of **Unity Editor (2021.2.0** to ??): the Android plugin folder needs to be renamed. Change AgoraRtcEngineKit.plugin to AgoraRtcEngineKit.androidlib. Note that this change is not compatible with earlier version of the Unity (e.g. 2017.x). Leave other settings as is. Version 2022.2.0 and up are confirmed that this behavior has been corrected, no manual renaming is needed.

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

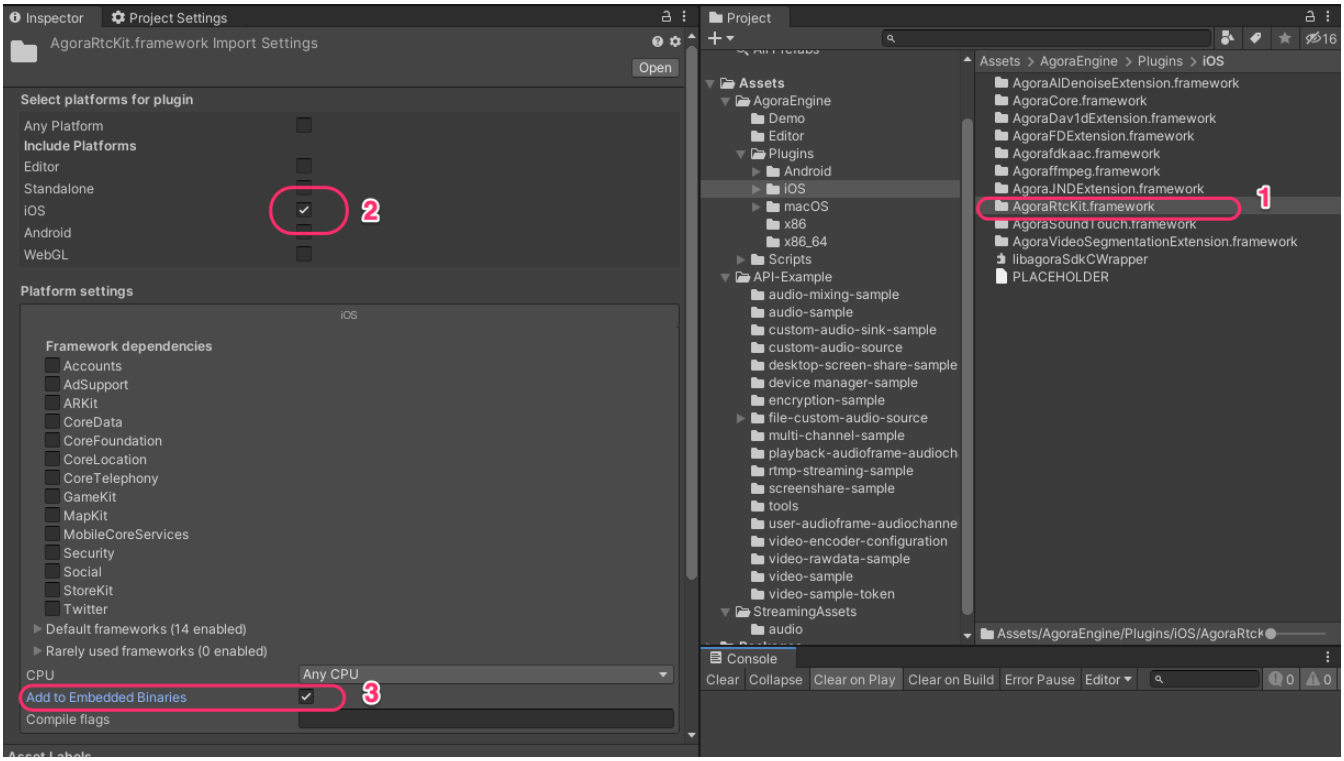
Embedding Frameworks

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 perform the following steps.

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:

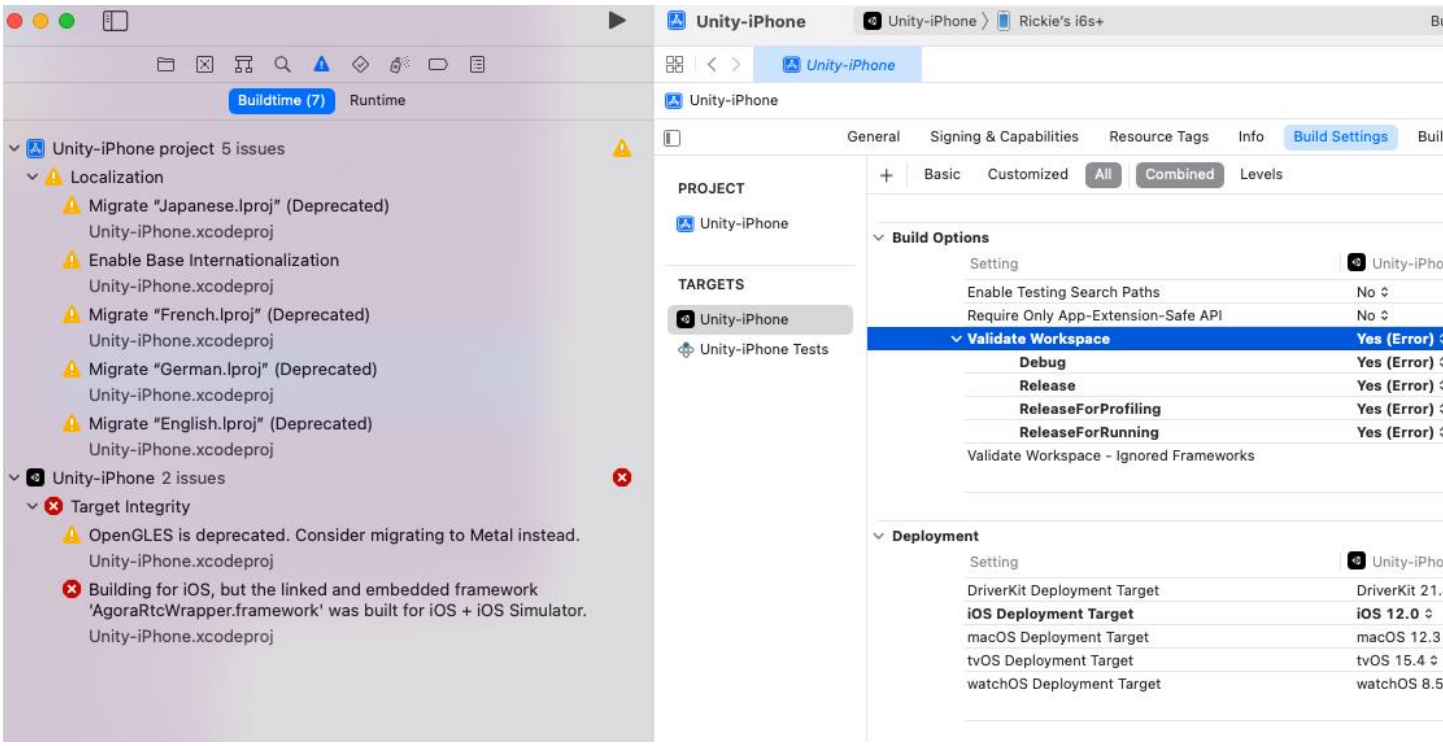
Make sure you do for all the items in the iOS folder:



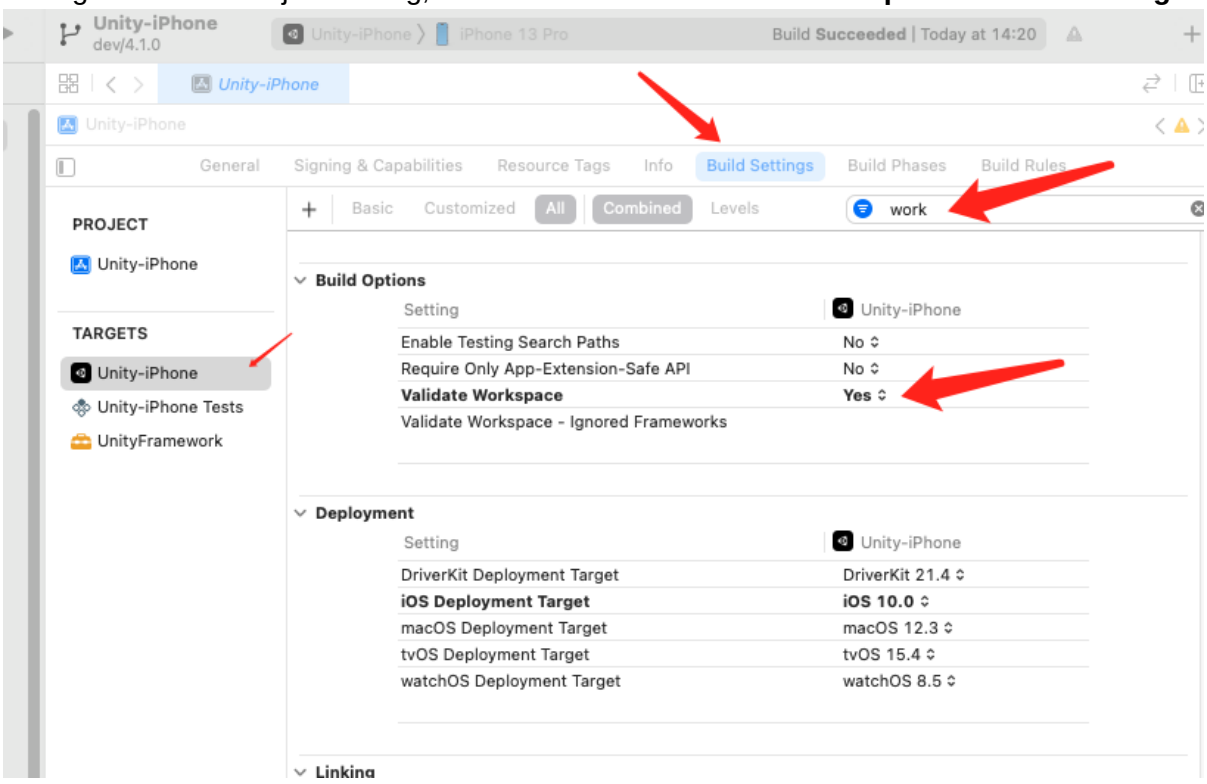
Unity Editor Embedded Binary setting (Unity 2019 and up)

Embedding Frameworks in V4.x

If there is a XCode linker error about “framework was built for iOS + iOS Simulator”, like this screen:

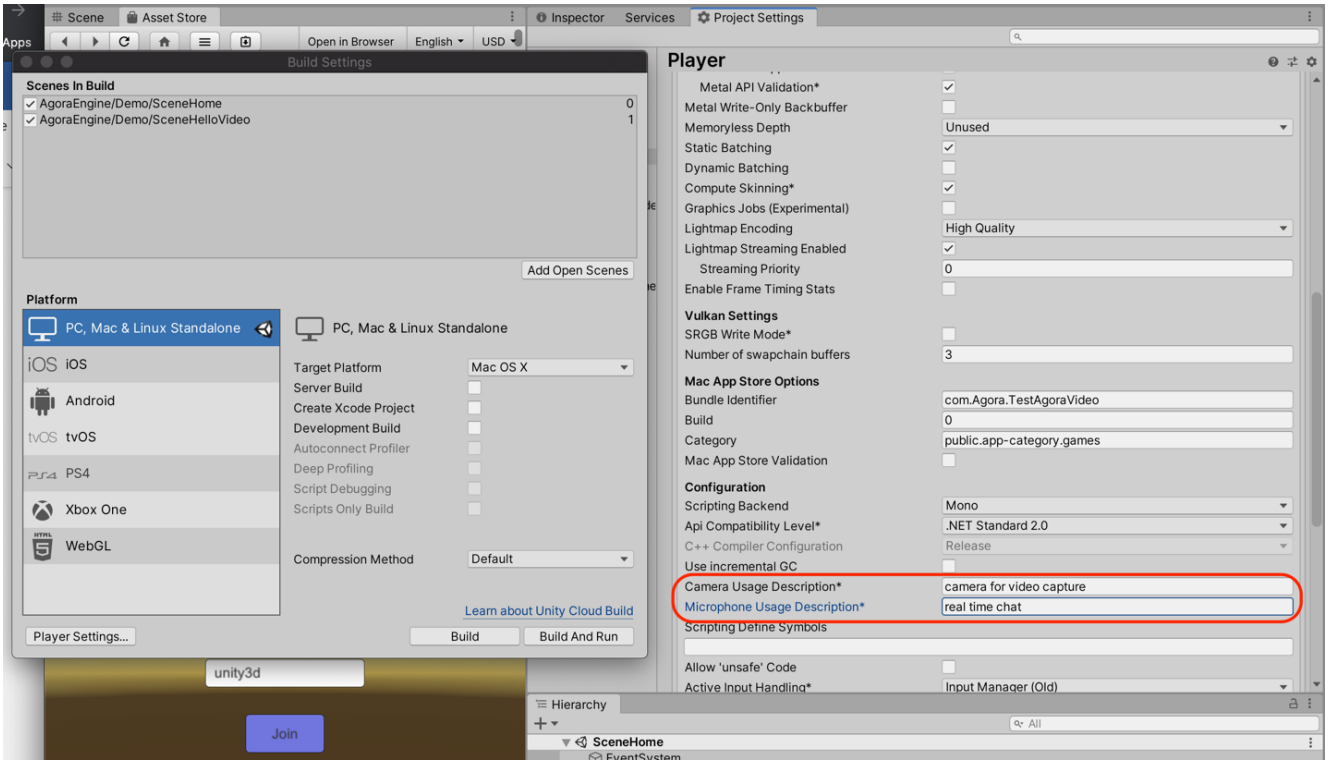


Configure XCode Project setting, choose **"Yes"** for **Validate Workspace** in **Build Settings:**



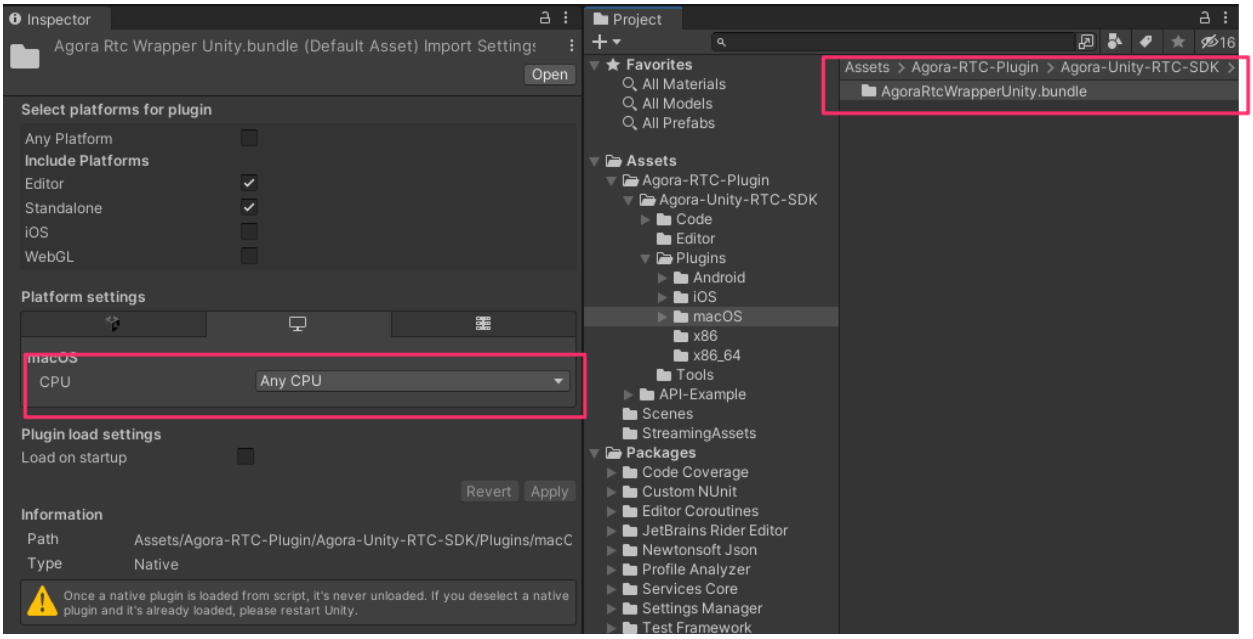
MacOS Build

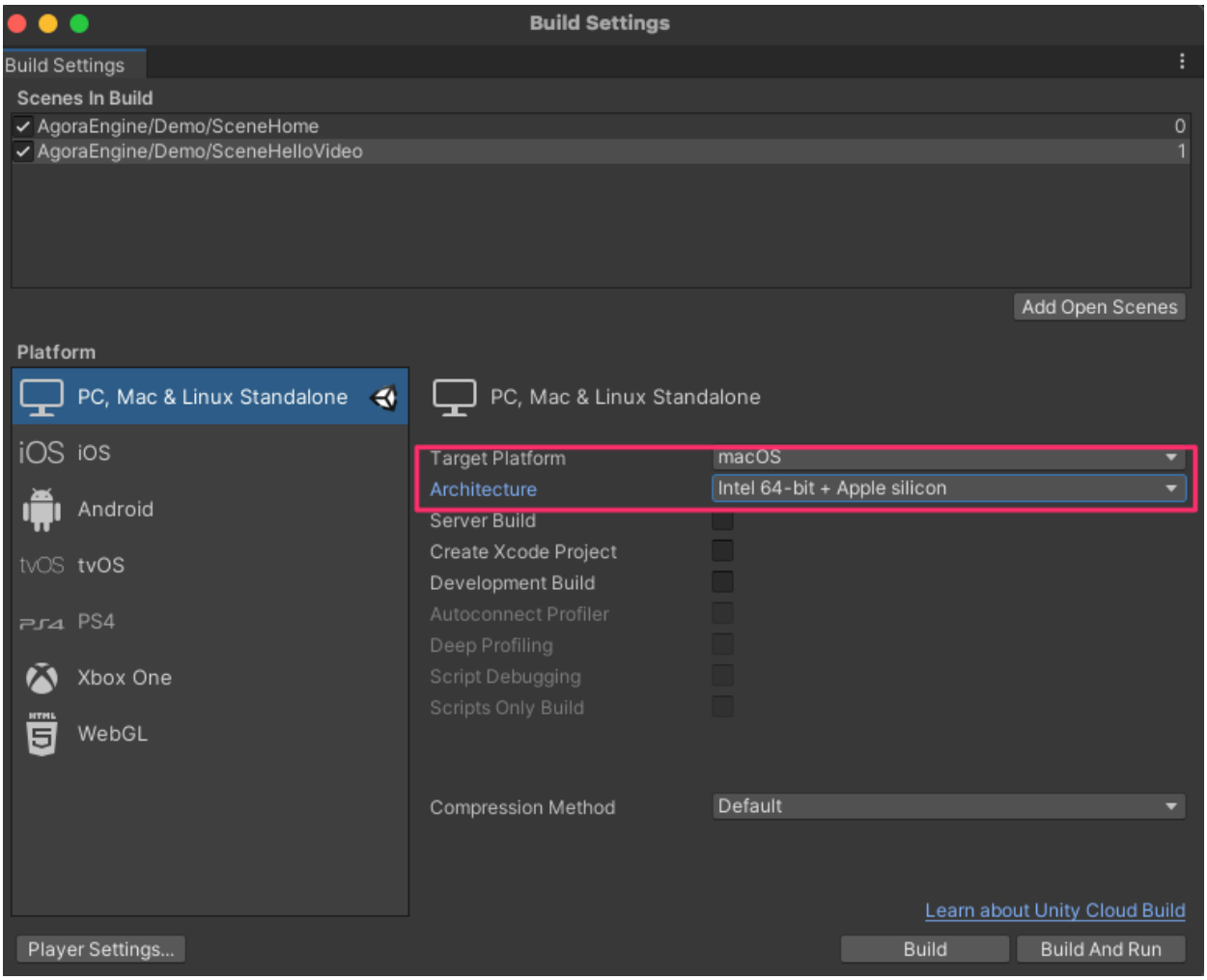
Select **“PC, Mac & Linux Standalone”** from the platform list and click **Switch Platform**. Then Select “**Mac OS X**” for the Target Platform. Besides regular App Store required information, it is **very important** to fill in the **Camera and Microphone** usage description. Without this your app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



MacOS Build Settings

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


Inspector
Agora Rtc Wrapper Unity.bundle (Default Asset) Import Setting:
Open
Select platforms for plugin
Any Platform
Include Platforms
Editor
Standalone
iOS
WebGL
Platform settings
macOS
CPU
Any CPU
Plugin load settings
Load on startup
Revert
Apply
Information
Path
Assets/Agora-RTC-Plugin/Agora-Unity-RTC-SDK/Plugins/macC
Type
Native
Once a native plugin is loaded from script, it's never unloaded. If you deselect a native plugin and it's already loaded, please restart Unity.
Project
Favorites
All Materials
All Models
All Prefabs
Assets
Agora-RTC-Plugin
Agora-Unity-RTC-SDK
Code
Editor
Plugins
Android
iOS
macOS
x86
x86_64
Tools
API-Example
Scenes
StreamingAssets
Packages
Code Coverage
Custom NUnit
Editor Coroutines
JetBrains Rider Editor
Newtonsoft Json
Profile Analyzer
Services Core
Settings Manager
Test Framework
Assets > Agora-RTC-Plugin > Agora-Unity-RTC-SDK >
AgoraRtcWrapperUnity.bundle


Build Settings
Build Settings
Scenes In Build
AgoraEngine/Demo/SceneHome
0
AgoraEngine/Demo/SceneHelloVideo
1
Add Open Scenes
Platform
PC, Mac & Linux Standalone
iOS
Android
tvOS
PS4
Xbox One
WebGL
PC, Mac & Linux Standalone
Target Platform
macOS
Architecture
Intel 64-bit + Apple silicon
Server Build
Create Xcode Project
Development Build
Autoconnect Profiler
Deep Profiling
Script Debugging
Scripts Only Build
Compression Method
Default
Learn about Unity Cloud Build
Player Settings...
Build
Build And Run

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/Agora-RTC-Plugin/Agora-Unity-RTC-SDK/Tools/prep_codesign.sh TestMAC.app
3. find the signature on the Mac with this command /usr/bin/security find-identity -v -p codesigning （there is a hint from the script also）
4. (export SIGNATURE="<your signature>"; ${project_path}/Assets/ Agora-RTC-Plugin/Agora-Unity-RTC-SDK/Tools/signcode.sh TestMAC.app)
5. Verify that the result

```
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.

Remember to set the appropriate Plugin library identities as described in the earlier section of this README file.

Upgrading SDK

Please check the online Migration guide when upgrading from Version 3.x to 4.x. Also, it is recommended to:

- Remove the entire file tree under Assets/AgoraEngine

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

Some Trouble Shooting FAQs

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: Do you support **WebGL**?
A: No, not in this version. But it is on the roadmap. Please come to our Slack channel and ask about the open beta program.

Q: Can I share screen with my camera feed too?
A: Yes, it is now supported by this SDK. See the DualCamera API example for reference project.

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

Other Resources

- MagicLeap2 Support: https://github.com/AgoraIO-Extensions/Agora_MagicLeap2_Plugin
- Agora Signaling SDK: https://docs.agora.io/en/signaling/overview/product-overview?platform=unity

- Agora Chat SDK: https://docs.agora.io/en/agora-chat/overview/product-overview?platform=unity
- GitHub demos: https://github.com/AgoraIO-Extensions/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/39j4I5h
- Help each other at https://agoraiodev.slack.com/messages/unity-help-me
- Developer relations team: devrel@agora.io
- Release note: https://docs.agora.io/en/video-calling/reference/release-notes?platform=unity