HISPlayer Unity SDK

HISPlayer is the most advanced video streaming player for Unity, supporting local, VOD and live content. It enables premium DASH and HLS video streaming inside your games, metaverses, and VR/AR apps. We include advanced features such as multiple streams or automatic bitrate adaptation to secure the best video quality.

You can find our **complete online documentation** here: HISPlayer Unity SDK

This SDK supports the following platforms:

- Android
- iOS
- WebGL
- macOS
- Windows
- UWP (Universal Windows Platform)

Supported Unity versions:

- 2020
- 2021
- 2022
- 2023
- Unity 6

QuickStart Guide

Getting started with HISPlayer consists of implementing the following steps:

1. Import and configure package

   1.1. Import HISPlayerSDK package

   1.2. Configure Unity for different platforms

2. Create your own sample

   2.1 Setup HISPlayer Manager

   2.2 Attach Unity Resources

   2.3 Configure HISPlayer Properties

   2.4 Build and Run

It's also possible to use the **HISPlayer Sample** after completing step 1 and skip step 2. The sample is a comprehensive example scene using the HISPlayerSDK to help demonstrate features like play, pause, seek, etc. Please refer to the Import HISPlayer Sample section.

1. Import and Configure Package

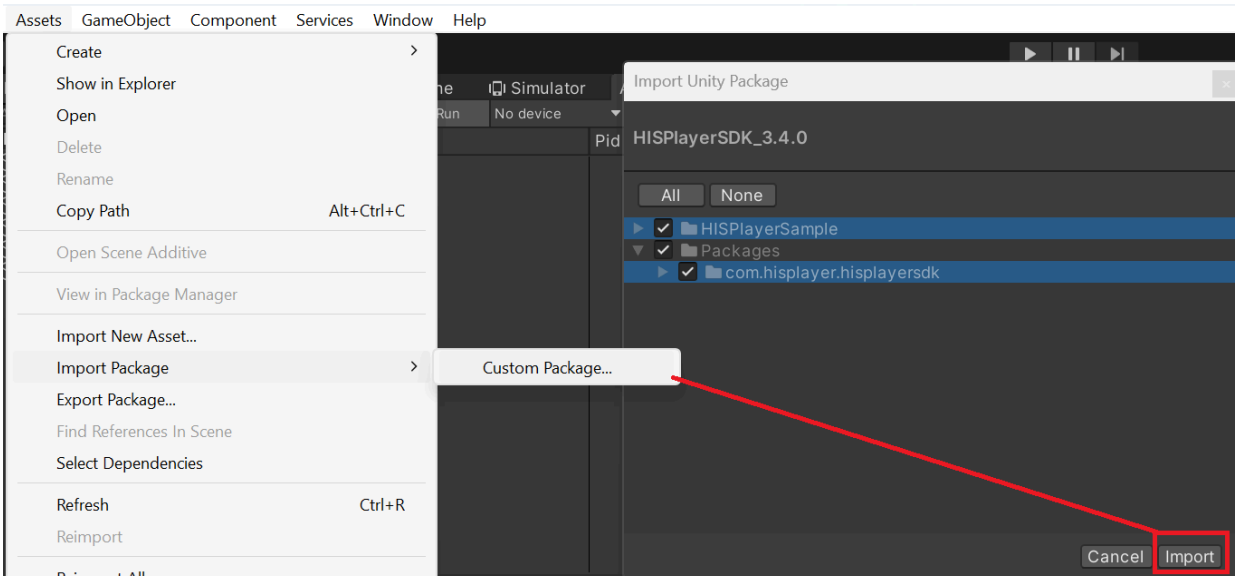
1.1. Import HISPlayerSDK Package

The HISPlayer Sample is included in the HISPlayer SDK Unity Package.

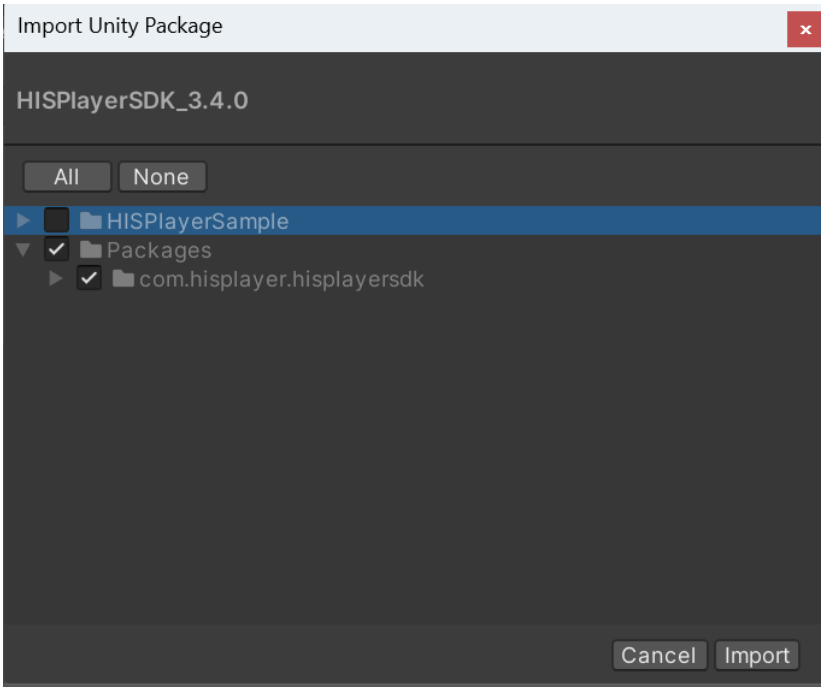
Importing the package is the same as importing other normal packages in Unity. Select the package of HISPlayer SDK and import it.

- **Assets > Import Package > Custom Package > HISPlayerSDK unity package**

Refer to HISPlayer Sample to know more about the HISPlayer Sample.



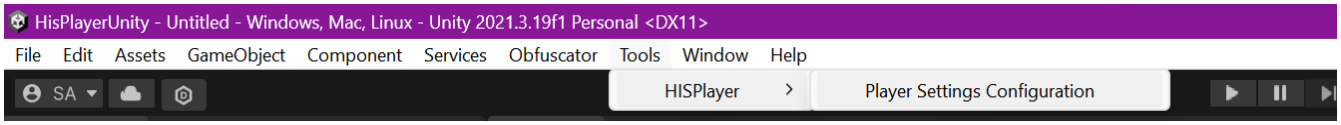
In the case you don't want to include the HISPlayer Sample, please disable it from the Import Unity Package window.



1.2. Configure Unity for different platforms

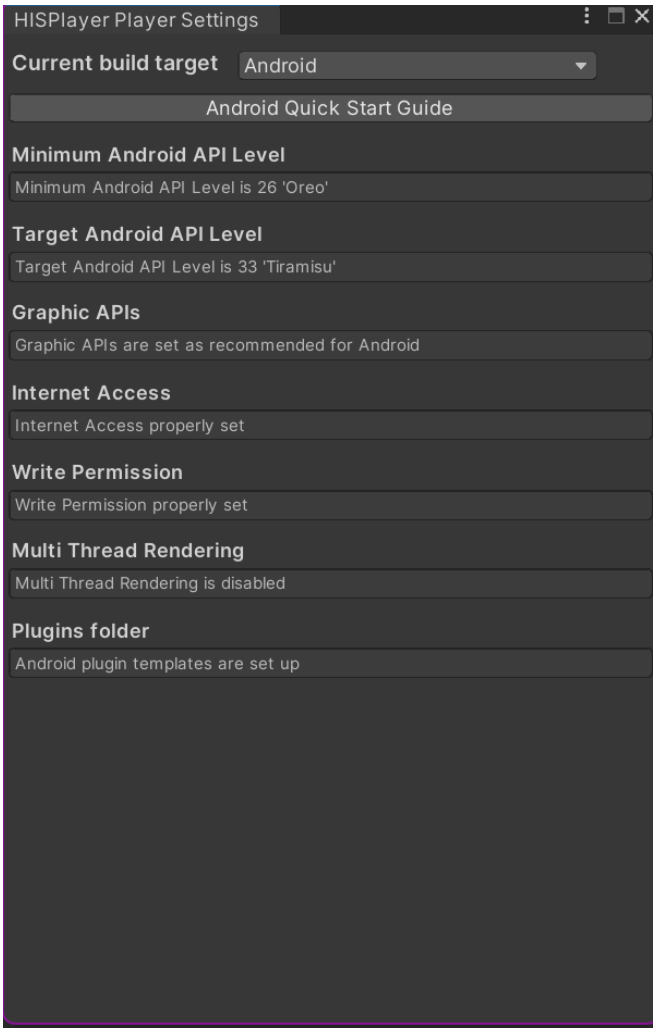
Open the window **Tools > HISPlayer** located in the upper side of the screen, Click on **Player Settings Configuration > Select the desired Build Target > Set all the required settings**.

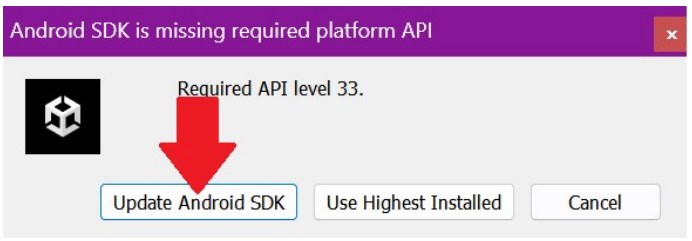
Depending on the target you will need to do some extra configuration.



Android

Set all the required settings. By selecting **Android target 33**, Unity is going to ask you to update the Android SDK (in the case you don't have the SDK 33 installed). Please, press the "Update Android SDK" button.

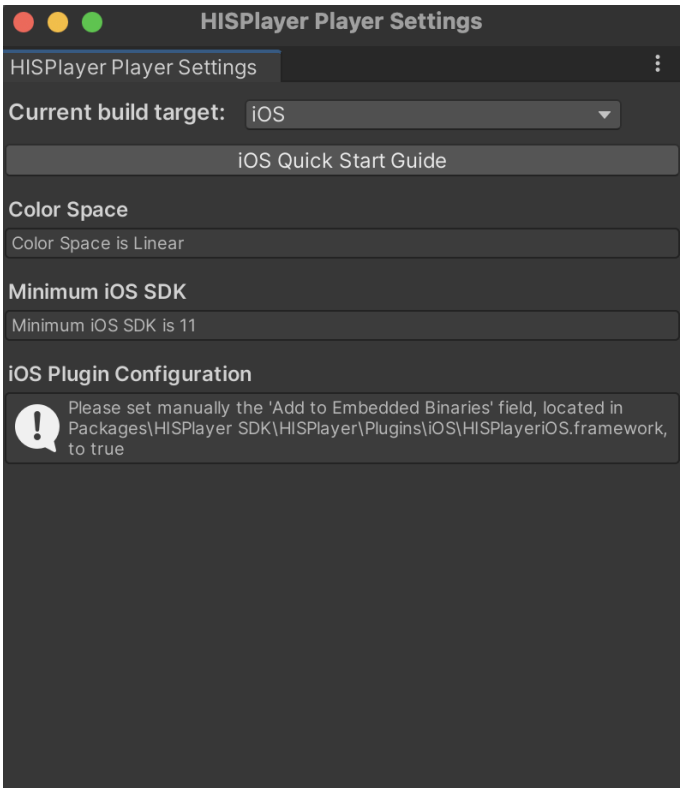


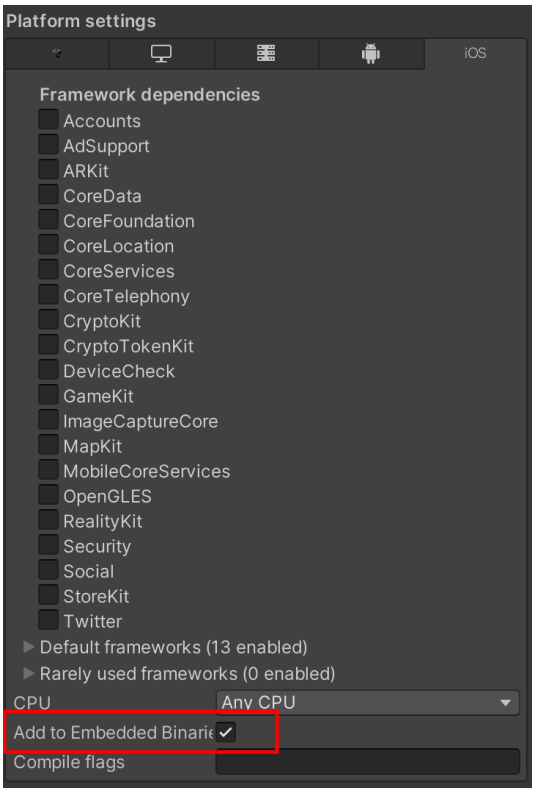


iOS

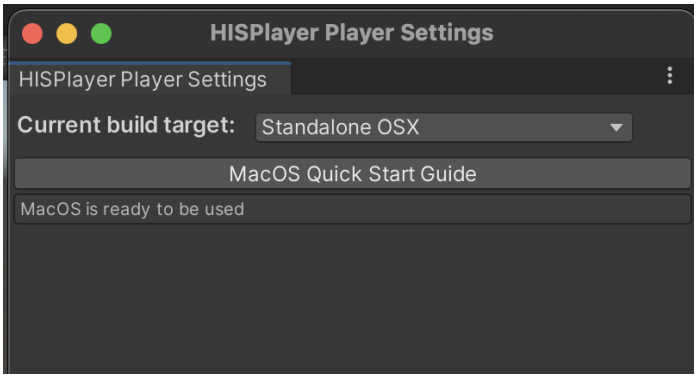
The native plugin for iOS needs theAdd to Embedded Binaries to be checked.

- **Packages > HISPlayer SDK > HISPlayer > Plugins > iOS**: Click the **HISPlayeriOS.framework** and check the Inspector. Select the plugin platform for **iOS**. Make sure that the HISPlayeriOS.framework file has the **Add to Embedded Binaries** set to true.





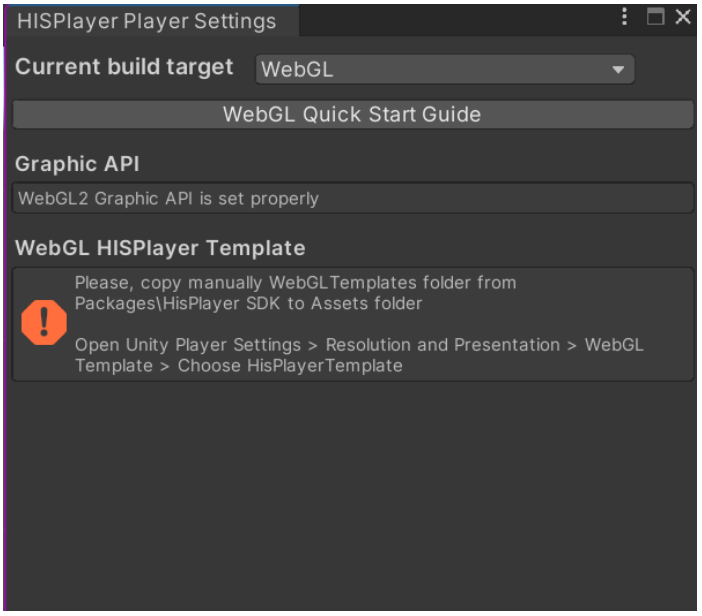
macOS



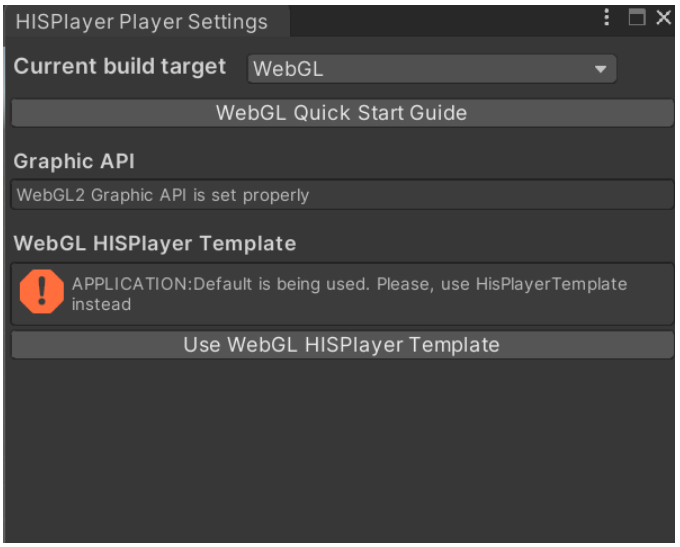
WebGL

The **Packages > HISPlayer SDK > WebGLTemplates** must be **copied manually** to the Unity Assets folder.

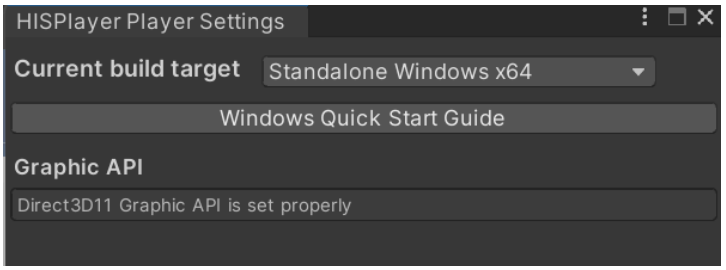
In the case you already have a WebGLTemplates folder in the Assets folder, copy **Packages > HISPlayer SDK > WebGLTemplates > HisPlayerTemplate** into your WebGLTemplates folder.



Finally open **HISPlayer Player Settings** again to use the HisPlayerTemplate

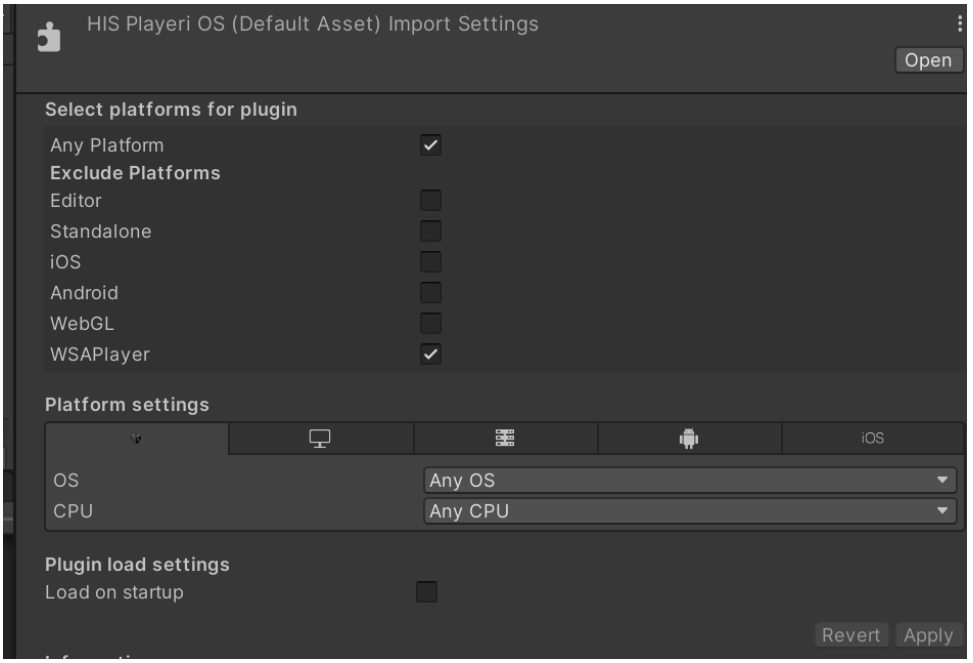


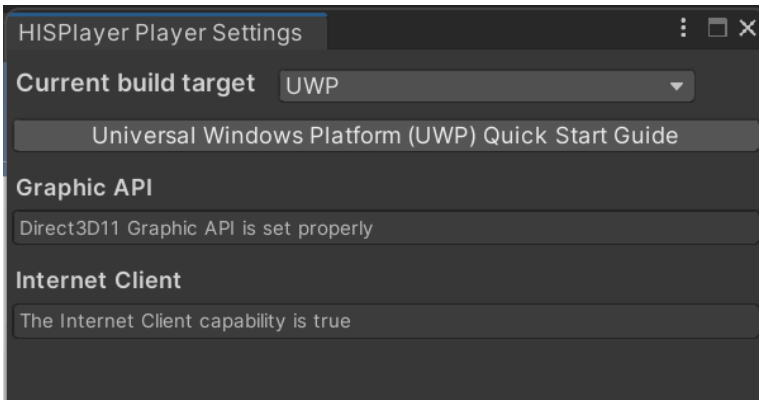
Windows



UWP

In the Project window, open **Packages > HISPlayer SDK > HISPlayer > Plugins > iOS** select **HISPlayeriOS.framework** and **exclude WSAPlayer** platform.





In the case you are using Unity 2022.1 or above, please, enable **Player Settings > Publishing Settings > Sync Capabilities**.

Import HISPlayerSample

The HISPlayer Sample is included in the HISPlayer SDK Unity Package. Please, refer to:

- 1.1. Import HISPlayerSDK Package

You can find the list of other sample packages in the following documentation:

- https://hisplayer.github.io/UnitySamples/#/

How to use HISPlayer Sample

To use the sample, please follow these steps:

- Make new Unity project
- Import HISPlayerSDK package (HISPlayer Sample included)
- Configure HISPlayer Settings, please refer to the previous section
    - Android
    - iOS
    - WebGL
    - macOS
    - Windows
    - UWP
- Open HISPlayerSample/Assets/Scenes/HISPlayerSample.unity
- Import TextMesh Pro Essential
- Open File > Build Settings > Add Open Scenes
- Build and Run

To check how to set up the SDK and API usage, please refer to:

- **Assets > HISPlayerSample > Scripts > Sample > HISPlayerSample.cs and HISPlayerController GameObject**

You may modify the sample following your needs.

2. Create Your Own Sample

You may skip this step if you have imported the HISPlayerSample package.

2.1 Setup HISPlayer Manager

Create a new script which will inherit from **HISPlayerManager**, for example, HISPlayerStreamController. It is necessary to add the **'using HISPlayerAPI;'** dependency. Then, add this component to a new game object (recommended to be empty).

Call the **SetUpPlayer()** function in order to initialize the stream environment internally. This function can be called whenever it's needed. For example, using the Awake function:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using HISPlayerAPI;

public class HISPlayerStreamController : HISPlayerManager
{
   protected override void Awake()
   {
      base.Awake();
      SetUpPlayer();
   }
}
```

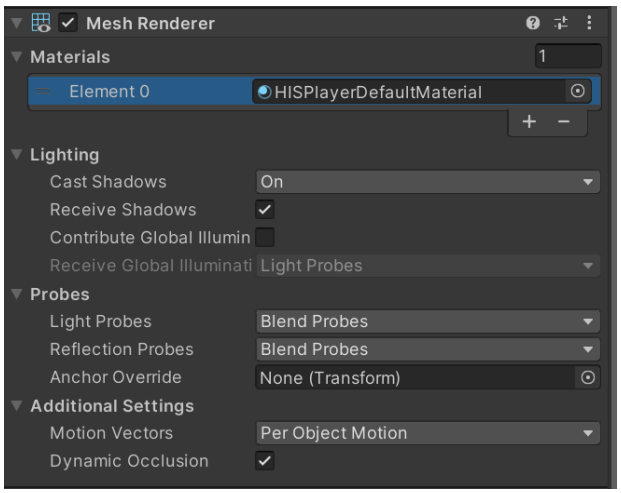
It is strictly necessary to use **SetUpPlayer** before using anything else. This function initializes everything that will be needed during the usage of HISPlayer APIs. Remember to call the **Release** function after closing the app or before changing scenes in Unity for freeing the internal resources.

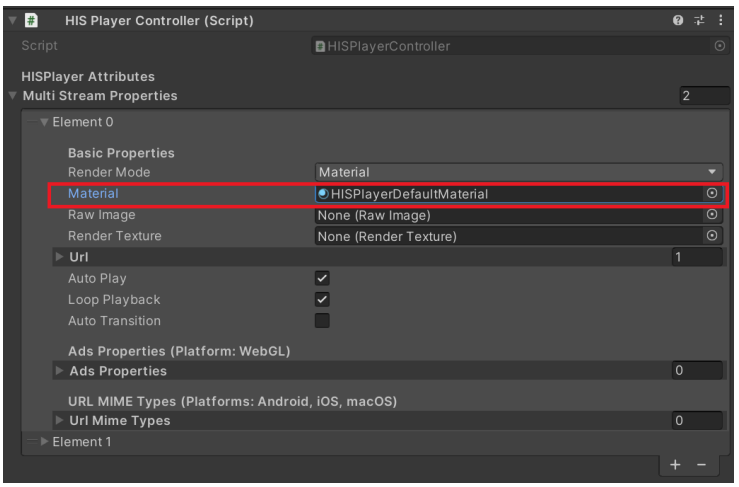
2.2 Attach Unity resources

Move to **Unity Editor** to attach all the resources. The rendering system is supporting **Material, RawImage** and **RenderTexture** Unity's components.

Material

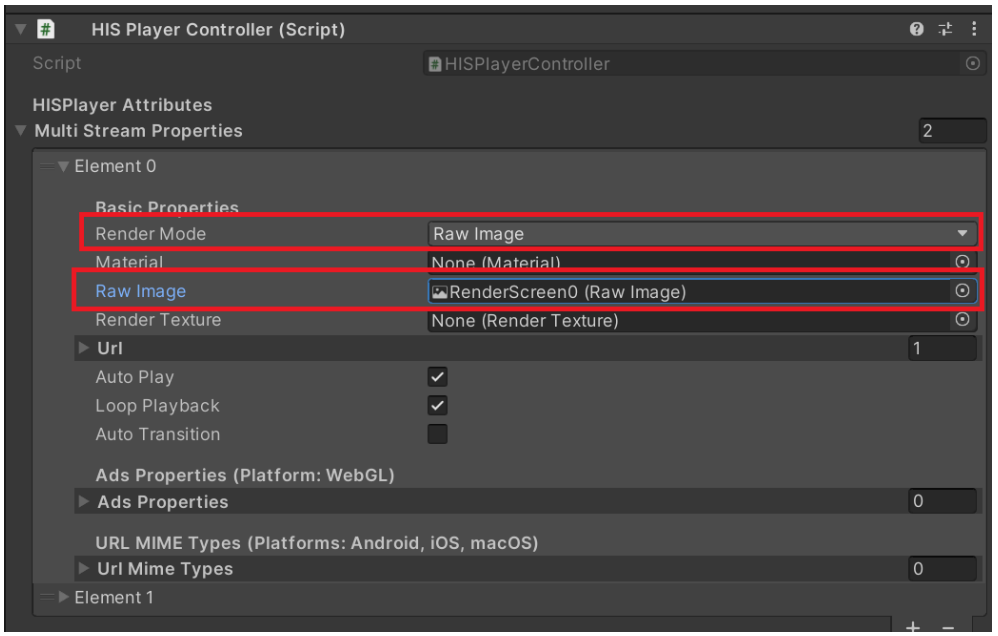
Create a new Material from **Assets > Create > Material** and attach it to the GameObject that is going to be used as screen and to the stream controller component. You can also use the **Packages > HISPlayer SDK > Resources > Materials > HISPlayerDefaultMaterial.mat** we provide in our package.





Raw Image

This action will be related to Unity's Canvas. If there is not a Canvas created yet, creating a **Raw Image** will create one automatically. For the creation, select **GameObject > UI > Raw Image**. Once it is created, attach it to the stream controller component

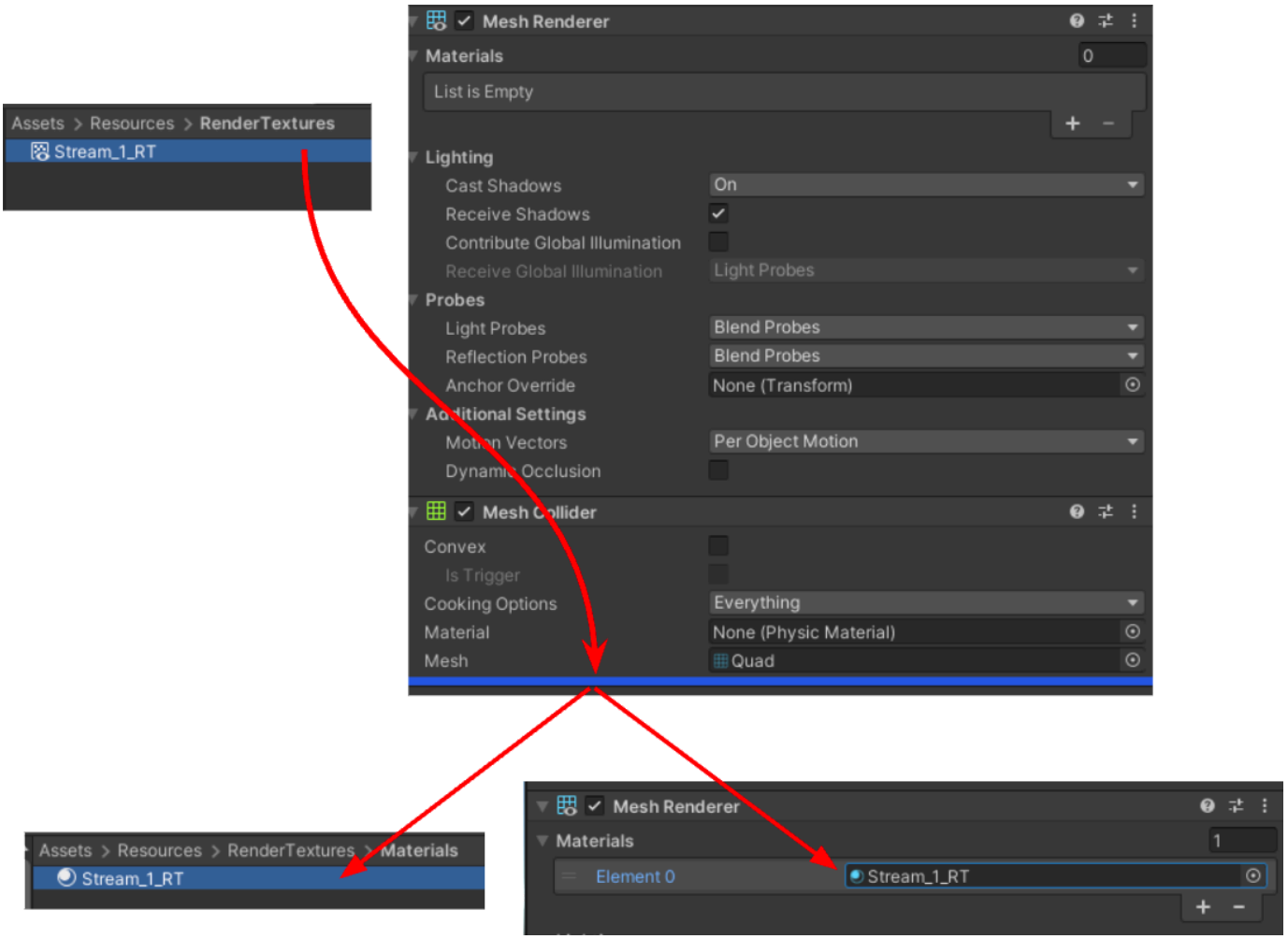


RenderTexture

For this you can use the RenderTexture we provide or create a RenderTexture from zero. In the first case, go to the Resources folder of our package and attach the **Packages > HISPlayer SDK > Resources > Materials > HISPlayerDefaultMaterialRenderTexture.mat** to the GameObject that is going to be used as screen and the **Packages > HISPlayer SDK > Resources > RenderTextures > HISPlayerRenderTexture.renderTexture** to the StreamController component.

For creating it from zero, select **Assets > Create > Render Texture** and then create a **Material** referencing the **Render Texture**. This last action can be done automatically by grabbing the **Render Texture** and dropping it at the end of a GameObject's Inspector with the component **Mesh Renderer** with **Material field empty**. This will create the new material inside a Materials folder.

Once all this process it's done, associate the **RenderTexture** to the script component.

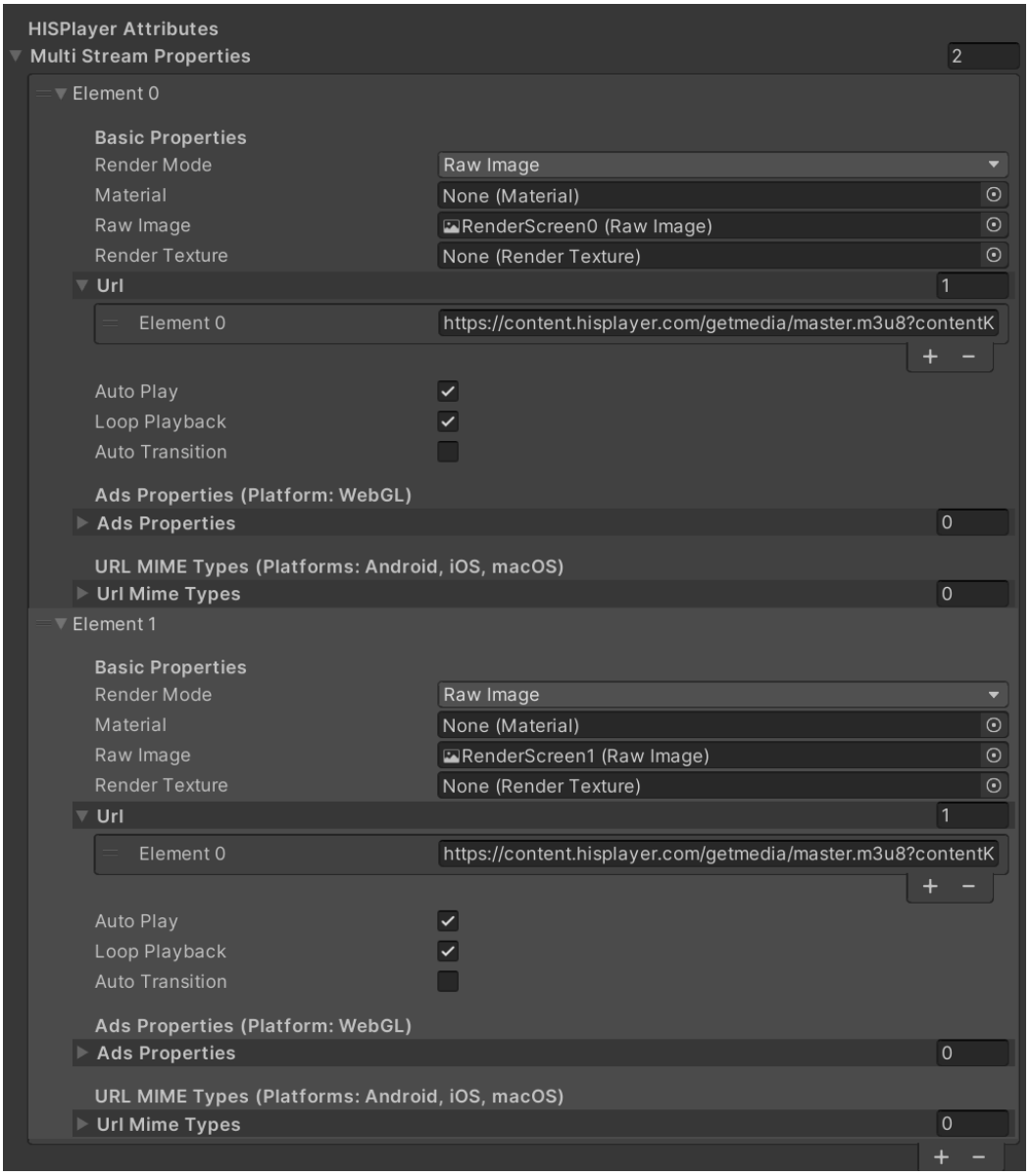


2.3 Configure HISPlayer Properties

Use Multi Stream Properties to set all the configuration needed for multi stream. It starts with 0 elements. Each element added has its own configuration for multiple players and corresponds to 1 Render Surface. If you just need a single stream, then you just need to add 1 element with 1 URL.

- **Render Mode**: Select the render surface. It can be RenderTexture, Material, RawImage or NONE.
- **Material**: Attach the Material asset created to the Material section of the element.
- **Raw Image**: Attach the created RawImage asset to the RawImage section of the element.
- **Render Texture**: Attach the RenderTexture to the RenderTexture section of the element.
- **URL**: Add the URL associated to the stream. Each stream can have multiple URLs, therefore users can use the same render surface to play different URLs. It is also possible to add local files allocated in the device's storage and the StreamingAssets special folder of Unity.

- **Autoplay**: Property to determine whether the player will start automatically after set up.
- **Loop Playback**: Property to loop the current playback. It's true by default.
- **Auto Transition**: Property to change the playback to the next video in the playlist. This action won't have effect when loopPlayback is true. It's false by default.
- **URL MIME Types**: Set the MIME types of each URL. It can be using URL Extension, HLS or DASH. URL Extension is set by default. **Supported platforms: Android, iOS, macOS**
- **Ads Properties**: List of properties to configure advertisement insertions for each player in the scene. **Supported platforms: WebGL**



2.4 Build and Run

Once the configuration it's done, open **Build Settings** and press **Build And Run**.

HISPlayer API

You can find the full information about HISPlayer API for each platform in the following links:

- Android
- iOS
- WebGL
- macOS
- Windows + UWP

Update the SDK

To update the package if you have the previous version of HISPlayer SDK, please follow these steps:

- Remove the previous HISPlayer SDK package from Unity Package Manager
    - **Window > Package Manager > Packages - HISPlayer > HISPlayerSDK > Remove**
- Import the new **HISPlayerSDK** package
- Configure HISPlayer Settings, please refer to the previous section
    - Android
    - iOS
    - WebGL
    - macOS
    - Windows
    - UWP

Oculus / Meta Quest

For **Oculus/Meta Quest** usages and samples, please refer to the below documentations:

- Using Meta XR All-In-One SDK
    - Import HISPlayer Meta XR Sample (you will find the download link for the sample package on this page)
- Using Oculus Integration SDK (Legacy)
    - Import HISPlayer Oculus Sample (you will find the download link for the sample package on this page)