HOLOLIGHT

HOLOLIGHT
STREAM

Hololight Stream SDK for Unity

Holo-Industrie 4.0 Software GmbH | Carl-Zeiss-Ring 19 | 85737 Ismaning

Table of Contents

Introduction 4

1. Set up Hololight Stream and additional packages 5

1.1. Hololight Stream setup 5

1.1.1. Hololight Stream system recommendations 5
1.1.2. Before upgrading from a previous version of Hololight Stream 7
1.1.3. Install Hololight Stream into your Unity project 7
1.1.4. Configure the project 8
1.1.5. Run Hololight Stream for the first time 8
1.1.6. Using Hololight Stream with other Unity packages 8
1.1.7. Disabling Hololight Stream 9
1.1.8. Troubleshooting 9
1.1.9. License 10

1.2. Hololight Stream examples setup 11

1.2.1. Install Hololight Stream examples into your Unity project 12
1.2.2. Add a Hololight Stream example to your project 12

1.3. Install XR Interaction Toolkit 13

1.4. Install Microsoft Mixed Reality Toolkit 13

1.4.1. Install Microsoft Mixed Reality Toolkit 2 14
1.4.2. Install Hololight Stream Extension for MRTK2 14
1.4.3. Install Microsoft Mixed Reality Toolkit 3 16

1.5. Install AR Foundation 19

1.5.1. Install AR Foundation 20
1.5.2. Example 20
1.5.3. Limitations 21

2. Features and use 22

2.1. Example Unity project 22

2.2. Connection details 22

2.3. DirectX 12 23

2.4. Hololight Stream settings 23

2.4.1. Signaling Port 23
2.4.2. Streaming bandwidth 24
2.4.3. Session log 24
2.4.4. Port range 25
2.4.5. Configuration file 25

2.5. Core components 25
2.5.1. Audio 25
2.5.2. Camera 27
2.5.3. Haptics 28
2.5.4. Microphone 30

2.6. Extensions 31
2.6.1. QR codes 31
2.6.2. Image tracking (2D) 34
2.6.3. Raycasts 35
2.6.4. Spatial anchors 35
2.6.5. Occlusion 44
2.6.6. Object stabilization 47
2.6.7. Scene understanding 52
2.6.8. Meshing 53
2.6.9. Gaze 58
2.6.10. Touch 60
2.6.11. Speech Recognition 61
2.6.12. Video passthough 64

2.7. Custom extensions 69
2.7.1. Signaling 69

3. Hololight Stream Client 72

3.1. Supported Hololight Stream Client devices 72
3.1.1. Apple iOS and iPadOS 72
3.1.2. Apple visionOS 73
3.1.3. HTC VIVE Focus 73
3.1.4. Lenovo ThinkReality VRX 74
3.1.5. Magic Leap 2 74
3.1.6. Meta Quest 75
3.1.7. Microsoft HoloLens 2 76
3.1.8. PICO 4 Ultra 76
3.1.9. Web browser 77
3.1.10. Windows 78

3.2. Connect to Hololight Stream application with Hololight Stream Client 78

3.3. Hololight Stream Client settings 79

Introduction

Hololight Stream is a powerful remote rendering solution designed to stream high-end XR experiences to a wide-variety of client devices. Run AR and VR applications on a local server or cloud-based infrastructure and stream them to supported headsets, smartphones, and tablets. Bring the power of a high-end server machine to a wide-variety of devices.

1. Set up Hololight Stream and additional packages

1.1. Hololight Stream setup

To use Hololight Stream you need to add it to your Unity project.

Hololight Stream official supports the following Unity versions:

- Unity 2022.3 (LTS)
- Unity 2021.3 (LTS)

NOTE	Hololight Stream may support newer versions of Unity, but they are not fully tested.

1.1.1. Hololight Stream system recommendations

Hololight Stream is able to support applications with a range of visual complexity and deliver compelling XR experiences to a wide-array of supported devices. On its own, Hololight Stream's requirements are minimal but scale up depending on the demands of the application you want to integrate Hololight Stream into. This guide outlines considerations you should make when choosing to develop with Hololight Stream.

Server recommendations

Components	Minimum	Good	High-end
Operating System	Windows 10 Windows 11	Windows 10 Windows 11	Windows 10 Windows 11
CPU	x64-based	x64-based	x64-based
Cores	4	8	16
RAM	16 GB	32 GB	64 GB
GPU[1]	NVIDIA graphics cards	NVIDIA graphics cards	NVIDIA graphics cards
VRAM	8 GB	10 GB	48 GB

[1] For more information regarding graphics card recommendations see **GPU recommendations**

GPU recommendations

We test Hololight Stream on workstations using NVIDIA graphics cards. The following recommendations are the cards which deliver the performance needed to create enterprise-level XR experiences powered by Hololight Stream.

The suggested graphics cards vary depending on two factors. One is the target client devices that will connect to your application using Hololight Stream Client. The second is the level of complexity of the application you are developing.

Low, **Medium**, and **High** represent different levels of application complexity. These are affected by factors like the number of polygons, graphics effects, and logic in the application. If your application is lightweight, with low complexity models, you can expect graphics cards in the **Low** column to work well. On the other hand, very complex applications will require graphics cards in the **High** column.

Hololight Stream server GPU recommendations

Target client device	Low	Medium	High
Apple iOS iPadOS	RTX 2080 RTX 3070	RTX A4000 RTX A4500 RTX A5000	RTX A6000
Apple Vision Pro	RTX 4070 Ti Super RTX 4080	RTX 4000 ADA	RTX A6000
HTC VIVE	RTX 4070 Ti Super RTX 4080	RTX 4000 ADA	RTX A6000
Lenovo ThinkReality VRX	RTX 4070 Ti Super RTX 4080	RTX 4000 ADA	RTX A6000
Magic Leap 2	RTX 3080 RTX 3080Ti RTX 4060	RTX A4000 RTX A4500 RTX A5000	RTX A6000
Microsoft HoloLens 2	RTX 2080 RTX 3070	RTX A4000 RTX A4500 RTX A5000	RTX A6000
Meta Quest 2 Meta Quest Pro Meta Quest 3	RTX 4070 Ti Super RTX 4080	RTX 4000 ADA	RTX A6000
PICO 4 Ultra	RTX 4070 Ti Super RTX 4080	RTX 4000 ADA	RTX A6000
Web browser application	RTX 2080 RTX 3070	RTX A4000 RTX A4500 RTX A5000	RTX A6000
Windows	RTX 2080 RTX 3070	RTX A4000 RTX A4500 RTX A5000	RTX A6000

1.1.2. Before upgrading from a previous version of Hololight Stream

If your project already has Hololight Stream installed, you need to remove any files left over from previous versions before installing the newest version.

1. In the Unity Editor menu bar select **Window > Package Manager**.
2. Find any Hololight Stream packages and select **Remove**.

When you've removed all files from older versions of Hololight Stream, you can continue on to installing the latest version.

1.1.3. Install Hololight Stream into your Unity project

1. Open your 3D Unity project.
2. Select **Window > Package Manager**.
3. In the Package Manger, select **+ > Add package from tarball**.
4. Find the Hololight Stream repository and open the "Packages" folder. Select the file "com.hololight.stream-<version>.tgz" where <version> is the current version of Hololight Stream.

NOTE	A warning may appear asking if you want to enable the native platform backends for the new input system. Select **Yes** and Unity will restart.

1.1.4. Configure the project

1. Select **Edit > Project Settings > XR Plug-in Management**.
2. Check the box next to **Initialize XR on Startup**.
3. Under **Plug-in Providers**, check the box next to **Hololight Stream**.
4. Select **File > Build Settings**
5. On the Build Settings panel, make sure to set the following options:

Platform	Windows, Mac, Linux
Target platform	Windows
Signaling Server	Intel 64-bit

1.1.5. Run Hololight Stream for the first time

After you finish setting up Hololight Stream, select **Play**. If you did everything correctly, you will now be able to connect to the application with Hololight Stream Client. Use the PC's IP address with Hololight Stream Client to connect to the application. When connected with Hololight Stream Client, you will see the contents of the first scene under **Scenes to Build**. If your scene is empty, you won't see anything. In Unity though, you will see the Main Camera move with the client device's movements.

You can also build and run a standalone application to test.

1.1.6. Using Hololight Stream with other Unity packages

With Hololight Stream in your Unity project, you can now add additional packages to extend Hololight Stream's features. You do not need these additional packages to use Hololight Stream. However, certain XR-specific features require one or more of the following:

- Mixed Reality Toolkit 2 *or* Mixed Reality Toolkit 3
- XR Interaction Toolkit
- AR Foundation

This guide includes details throughout explaining what features require which packages.

1.1.7. Disabling Hololight Stream

You can disable Hololight Stream without removing it with the Package Manager.

1. In the Unity Editor menu bar select **Edit > Project Settings > XR Plug-in Management**.
2. Uncheck the box next to **Hololight Stream** to disable Hololight Stream.
3. Check **Unity Mock HMD**.

Hololight Stream is now disabled. For your application to work correctly, make sure it doesn't call any Isar or Isar-inherited classes. This includes Isar constructors.

1.1.8. Troubleshooting

The client won't connect to my application

- Check that you correctly entered the server's IP address.
- Check your firewall settings. You need to allow ports used by Hololight Stream to bypass the firewall. These are the ports you entered under **Hololight > Hololight Stream > Settings > Signaling Port** and **Connection Port Range**.

Hololight Stream Settings default ports

Port type	Port number of range
Signaling port	9999
Connection port Range	50100 - 50100

The Unity Editor crashes when trying to run your application

If the Unity Editor keeps crashing and Hololight Stream is included in the project but not required, follow the steps in **Disabling Hololight Stream** to try and debug the issue.

The Main Camera won't move when it is supposed to

If your project uses the High Definition Render Pipeline and the Main Camera is not moving, remove the "SimpleCameraController" script from the Main Camera and try again. If you are still having issues, check for any other scripts that might be moving the Main Camera.

1.1.9. License

When using Hololight Stream, you agree to the proprietary license terms and conditions.

1.2. Hololight Stream examples setup

The Hololight Stream examples package is a collection of Unity examples showing you how to use the many Hololight Stream components.

Hololight Stream examples

Example	Description
Anchoring Sample (MRTK 2)	Demonstrates the anchoring functionality. Only available when connected to the HoloLens 2 client. Requires MRTK 2.
Anchoring Sample (MRTK 3)	Demonstrates the anchoring functionality. Only available when connected to the HoloLens 2 client. Requires MRTK 3.
Audio Stream Sample	Contains prefabs and scripts for streaming Unity audio to the connected device.
Camera Stream Sample	Contains prefabs and scripts for receiving the clients camera stream within Unity. Only available when connected to the HoloLens 2 client.
Data Channel Sample	Contains scripts highlighting how to create a custom extension requiring a data channel.
Occlusion Sample	Demonstrates the Occlusion Extension usage. Only available in combination with the iOS client.
Microphone Sample	Contains prefabs and scripts for receiving the clients microphone stream within Unity.
Signaling Sample	Contains scripts for providing external signaling implementation.
Disconnection Sample (MRTK 2)	Contains scripts for remote disconnection.Requires MRTK 2.
Disconnection Sample (MRTK 3)	Contains scripts for remote disconnection. Requires MRTK 3.
Object Stabilization Sample	Demonstrates how to set the focus plane to an object in order to provide better stability.
URP Alpha Passthrough	Sample code to allow alpha rendering in URP scenes in order to use device passthrough mode.
Meta Haptics	Sample code that demonstrates the usage of haptic files created by Meta Haptics Studio.
Eye Tracking	Sample code that demonstrates the usage of eye tracking information through the Input System.
Pose Prediction Configuration	Sample editor script that allows the tuning of the pose prediction.
Logitech MX Ink Stylus	Sample code demonstrating the usage of actions from Logitech MX Ink Stylus.
Passthrough	Sample code that demonstrates the usage of passthrough functionality on the server.

1.2.1. Install Hololight Stream examples into your Unity project

1. Select **Window > Package Manager**.
2. In the Package Manger, select **+ > Add package from tarball**.
3. Find the Hololight Stream repository and open the "Packages" folder. Select the file "com.hololight.stream.examples-<version>.tgz" where <version> is the current version of Hololight Stream.

After a little bit of loading, "Hololight Stream Examples" will appear under the category "Packages -Holo-Light GmbH".

1.2.2. Add a Hololight Stream example to your project

1. Select **Window > Package Manager > Hololight Stream Examples**.
2. Find the example you want and select **Import**.

1.3. Install XR Interaction Toolkit

The XR Interaction Toolkit is a package for creating VR and AR applications that leverages Unity input events for 3D and UI interactions.[2] XR Interaction Toolkit allows for number of XR features to work with your Hololight Stream application.

For more information on the XR Interaction Toolkit, check out Unity's **XR Interaction Toolkit documentation**.

For instructions on how to add XR Interaction Toolkit to your project. Check out Unity's **XR Interaction Toolkit Installation** page for more details.

1.4. Install Microsoft Mixed Reality Toolkit

Microsoft Mixed Reality Toolkit (MRTK) is a project that provides a set of components and features for accelerating XR application development in Unity.[3]

If you already have Mixed Reality Toolkit 2 or Mixed Reality Toolkit 3 installed in your project, skip to either **Configure the scene to use Mixed Reality Toolkit 2** or **Configure the scene to use Mixed Reality Toolkit 3**.

[2] "XR Interaction Toolkit: XR Interaction Toolkit: 3.0.7." XR Interaction Toolkit | 3.0.7. Accessed February 5, 2025. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html.

[3] "MixedRealityToolkit-Unity". GitHub. Accessed February 5, 2025. https://github.com/microsoft/MixedRealityToolkit-Unity.

1.4.1. Install Microsoft Mixed Reality Toolkit 2

Hololight Stream works together with Microsoft Mixed Reality Toolkit 2 (MRTK2) to make more immersive mixed-reality experiences. Follow the instructions below to add MRTK2 to your Unity project.

1. Go to the **Microsoft Mixed Reality Toolkit git project**.
2. Follow the instructions to download and install MRTK2. Hololight Stream supports MRTK2 version 2.8.3.

1.4.2. Install Hololight Stream Extension for MRTK2

Hololight Stream Extension for MRTK2 provides XR functionality for inputs and other features that makes using MRTK2 with Hololight Stream possible.

NOTE	MRTK2 will not work with Hololight Stream without the Hololight Stream Extension for MRTK2.

1. Select **Window > Package Manager**.
2. In the Package Manger, select **+ > Add package from tarball**.
3. Find the Hololight Stream repository and open the "Packages" folder. Select the file "com.hololight.stream.mrtk-<version>.tgz" where <version> is the current version of Hololight Stream.

Configure the scene

With MRTK installed in your Unity project, select **Hololight > Stream > Configure MRTK**, to configure the scene.

Check configuration

After you configure your project and scene, select **MixedRealityToolkit** in the **Hierarchy** and check the following settings:

1. Select **Camera > Camera Settings Providers**. Make sure **XR SDK Camera Settings** is one of the providers.
2. Select **Input > Input Data Providers**. Make sure that you have what input providers you want. These can include:
    - Hololight Stream Device Manager
    - Hololight Stream Touch Device Manager
    - Hololight Stream Dictation Input
    - Hololight Stream Speech Input
    - Handjoint service

Configure gaze

Some features are not enabled through scene configuration. The reason for this, is that gaze can conflict with other inputs. You will need to enable eye gaze functionality manually if you need this feature.

1. Select **MixedRealityToolkit** in the **Hierarchy**.
2. Select **Input** in the **Inspector**.
3. Under **Input Data Providers**, add Hololight Stream Eye Gaze Provider
4. Select **MixedRealityToolkit > Input > Pointers > Gaze Provider Settings**.
5. Make sure the box next to **Eye Tracking Enabled** is checked.

Test Mixed Reality Toolkit 2

Enter **Play** mode to test if you successfully configured MRTK2.

1.4.3. Install Microsoft Mixed Reality Toolkit 3

Hololight Stream works together with Microsoft Mixed Reality Toolkit 3 (MRTK3) to create more immersive mixed-reality experiences. Follow the instructions below to add MRTK3 to your Unity project.

IMPORTANT	If a project already has Mixed Reality Toolkit 2 (MRTK2) installed, you need to remove it completely before installing MRTK3. This includes Hololight Stream Extension for MRTK2. After completely removing these, you can install MRTK3 and the Hololight Stream MRTK3 extension. For more details about upgrading an MRTK2 project to MRTK3, check out Microsoft's page, **Migration guide from MRTK2 to MRTK3**.

1. Go to url: **https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/getting-started/setting-up/setup-new-project**
2. Follow Microsoft's instructions to download and install MRTK3.

NOTE	Newer versions of MRTK3 may work, but are not officially supported. Hololight Stream officially supports MRTK3.xx

Install Hololight Stream Extension for MRTK3

Hololight Stream Extension for MRTK3 provides XR functionality for inputs and other features that makes using MRTK3 with Hololight Stream possible.

To install Hololight Stream Extension for MRTK3 import "com.hololight.stream.mrtk3-<version>.tgz" where <version> is the latest version of Hololight Stream using Package Manager. This package is in the "Packages" folder of the Hololight Stream SDK.

NOTE	MRTK3 will not work with Hololight Stream without the Hololight Stream Extension for MRTK3.

Configure the project

If you have already configured your project to use Hololight Stream, you can then do the additional project configuration needed to support MRTK3.

Go to **Edit > Project Settings > MRTK3 > Profile**.

Configure with the added Stream Subsystems. A sample profile can be found in the package.
Packages/com.hololight.stream.mrtk3/Runtime/Profiles/StreamMRTKProfile.asset

Configure the scene

Configuring the scene for Hololight Stream. Hololight will add necessary components for Stream to an existing MRTK XR Rig or a Main Camera and then do the necessary modifications or instantiate the Stream MRTK XR Rig for the scene to use. To do this, you need the Hololight Stream Extension for MRTK3.

Select **Hololight > Stream > Configure MRTK**.

Optional features

Touchscreen input
Add the **Touch Controller Prefab** as a child to the Stream **MRTK XR Rig > Camera Offset** GameObject to enable touchscreen input.

Mouse input for Hololight Stream Client for Windows

1. Add the MRTK Spatial Mouse Controller Prefab as a child to the **Stream MRTK XR Rig > Camera Offset** GameObject to enable input on Hololight Stream Client for Windows.
2. Select **Stream MRTK XR Rig > Camera Offset > MRTK Spatial Mouse Controller**. In the **Inspector** go to **XR Controller (Action-based) > Select Action** and select the **Stream Mouse/Select Asset Inputs Action** found in the package.

3. Select **Stream MRTK XR Rig > MRTK Spatial Mouse Controller > MRTK Spatial Mouse Interactor**. In the **Inspector** under **MRTK Spatial Mouse Interactor**, change the following properties to **Asset Inputs Actions** found in the package:

Property	Value
Mouse Move Action	Stream Mouse/MouseMove
Mouse Scroll Action	Stream Mouse/MouseScroll

4. Select **Stream MRTK XR Rig > MRTK Spatial Mouse Controller > SpatialMouseInteractor** in the Hierarchy. In the Inspector set **MRTK Spatial Mouse Interactor > Mouse Sensitivity** to "0.5".
5. Select **Stream MRTK XR Rig > MRTK Spatial Mouse Controller > SpatialMouseInteractor > MouseCursor > CursorVisual** in the Hierarchy. In the Inspector under **Sprite Renderer > Flip**, make sure there is a checkmark next to **X**.

NOTE	MRTK Spatial Mouse Controller has a different behavior than MRTK 2's spatial mouse. Stream passes mouse movements directly to the MRTK 3 through Input Actions. It is also possible to use these Input Actions for different spatial mouse controller implementations.

Speech input

Select **Stream MRTK XR Rig > MRTK Speech** in the hierarchy. Make sure the GameObject is enabled.

Gaze

To enable gaze: Select **Stream MRTK XR Rig > Camera Offset > MRTK Gaze Controler** in the Hierarchy. Under **XR Controller (Action-based with Fallbacks)** change the following properties to **Asset Inputs Actions** found in the package:

Property	Value
Position Action Stream Gaze/Rotation	Position Action Stream Gaze/Rotation
Tracking State Action Stream Gaze/TrackingState	Tracking State Action Stream Gaze/TrackingState

NOTE	Optionally, to customize gaze the reticle, add the **MRTK Ray Reticle Visual** component to **Stream MRTK XR Rig > Camera Offset > MRTK Gaze Controler > GazeInteractor**. Then add the Reticle prefab (or any other custom reticle Prefab) as a child to **GazeInteractor**. Then in **MRTK Ray Reticle Visual** reference the reticle Prefab.

Test Mixed Reality Toolkit 3

Enter **Play** mode to test if you successfully configured MRTK3.

1.5. Install AR Foundation

AR Foundation is a Unity package that "enables you to create multi-platform augmented reality (AR) apps with Unity. In an AR Foundation project, you choose which AR features to enable by adding the corresponding manager components to your scene. When you build and run your app on an AR device, AR Foundation enables these features using the platform's native AR SDK, so you can create once and deploy to the world's leading AR platforms."

Hololight Stream doesn't fully implement all AR Foundation features. Hololight Stream uses AR Foundation as a user-friendly way to expose data and functionality. This provides some, but not all, features in Hololight Stream.

Supported AR Foundation features by device

Feature	Apple Vision Pro	Apple iOS and iPadOS	HTC VIVE devices	Lenovo ThinkReality VRX	Magic Leap 2	Meta Quest devices	Microsoft HoloLens 2	Windows	Web Browser
2D and 3D body tracking									
2D image tracking		Yes							
3D Object tracking									
Camera		Yes					Yes		
Collaborative participants									
Device tracking	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Environment probes									
Face tracking									
Human segmentation									
Light estimation									
Meshing		Yes					Yes		
Occlusion		Yes							
Plane tracking		Yes					Yes		
Point clouds									

Feature	Apple Vision Pro	Apple iOS and iPadOS	HTC VIVE devices	Lenovo ThinkReality VRX	Magic Leap 2	Meta Quest devices	Microsoft HoloLens 2	Windows	Web Browser
QR code tracking					Yes		Yes		
Raycast		Yes							
Session management									
Spatial anchors		Yes				Yes	Yes		
Passthrough video									

1.5.1. Install AR Foundation

Unity imports AR Foundation when you add Hololight Stream to the project. You still need to configure AR foundation in order to take advantage of the available features.

1. In the Hierarchy, add an AR Session Origin object (**GameObject > XR > AR Session Origin**)
2. Expand the AR Session Origin object and select the **AR Camera**. Set the AR Camera background to have alpha value of "0", within the inspector window

1.5.2. Example

Unity has an example project to demonstrate how AR Foundation works.

You can find the project **here**.

NOTE	These example scenes use the legacy input system for touch input which is not currently supported. You may need to update scenes that uses touch to use the new input system.

1.5.3. Limitations

- Some AR Foundation features aren't supported by Hololight Stream. This means not every AR Foundation example scene will work correctly. When using the example project with Hololight Stream, stick to the scenes which demonstrate the features marked as supported in the table above.
- Hololight Stream does not support transparency. Transparent objects are displayed without transparency if they are directly overlaying the background.
- To make the background transparent, make it black with an alpha value of 0.

2. Features and use

2.1. Example Unity project

HelloStream is a simple example which displays the Hololight Stream logo when connected with Hololight Stream Client. The example allows the user to view the object and move around the environment. However, the example does not contain any input interaction.

NOTE	HelloStream has a dependency on MRTK. You need to install either MRTK 2 or MRTK 3 along with the corresponding Hololight Stream MRTK Extension to use the sample. See **Install Microsoft Mixed Reality Toolkit** for more information.

2.2. Connection details

To see information about your current Hololight Stream connection status, select **Hololight > Hololight Stream > Connection Details**. Here you can see the following:

Client details

Property	Description
Device	The device that the connected Hololight Stream Client is running on.
Reality type	The type of XR experience the client is using. This can be: ▪ Augmented Reality ▪ Passthrough Augmented Reality ▪ Virtual Reality
Version	Version number for the connected Hololight Stream Client.

Video stream configuration

Property	Description
Resolution	Resolution of the connected Hololight Stream Client.
Target frame rate	The target frame rate for the stream.
Bandwidth	Maximum bandwidth limit while streaming.
Codec	Preferred codec for encoding and decoding.
Depth buffer streaming	Whether depth buffer is enabled in the stream.

2.3. DirectX 12

Hololight Stream supports DirectX 12 beginning with Unity 2022.2.0a17. This was the first version of Unity to officially supports DirectX 12. Single Pass Instanced Rendering with DirectX 12 is only supported in Unity version 2023.1.0f1.

Supported Unity versions

Unity Version	DirectX 12	Unity Single Pass Instanced Rendering
Pre 2022.2.0a17		
2022.2.0a17	Yes	
2023.1.0f1	Yes	Yes
2023.1.0f	Yes	

2.3.1. Setup

To use DirectX 12 with Hololight Stream you need to set it as the primary graphics API in your Unity project.

- Select **Edit > Project Settings > Player**.
- Under **Graphics APIs for Windows** make sure **Direct3D12** is the first in the list.

For more information on DirectX in Unity, check out Unity's page **DirectX**.

2.4. Hololight Stream settings

To configure server-side settings, select **Hololight > Hololight Stream > Settings** in the Unity project for your application.

2.4.1. Signaling Port

Sets the port used for signaling. The port number must be between 1024 and 65535. If unset, the port defaults to 9999. Make sure that the port numbers used with Hololight Stream aren't used by other applications or programs on your network.

2.4.2. Streaming bandwidth

Specifies the maximum encoder bit rate. A higher value will result in a better image quality but at the cost of network performance. The recommended range is between 10,000 - 100,000 Kbps. If the value is set to "-1", the bit rate will take its value from the settings in the connected client.

2.4.3. Session log

When enabled the application will log connection-specific information during an active connection. This is helpful when trying to diagnose connection issues. Hololight Stream stores the data as a CSV file at "<application_root_dir>/hls-stats/". You can use Excel to easily view the data.

You should only enable the session log when investigating issues since the logging will negatively impact performance. To enable this setting in a build application, specify stats-collector in the diagnostic-options section of the configuration file. The logs have the following data:

Timestamp
The time and date when the session log recorded an entry of data. The timestamp is an epoch.

Frame rate
The rate that the encoder receives frames.

Encoder target bit rate
The bit rate set for the encoder.

Available outgoing bit rate
The estimated bandwidth the network can reach.

2.4.4. Port range

The port range allows the user to specify the ports used for the running connection. The range must be between 1024 and 65535. You can set a single port by setting the minimum and maximum for the range as the same value. By default the port is set to 50100. This port can overlap with the signaling port.

Make sure that the port numbers used with Hololight Stream aren't used by other applications or programs.

2.4.5. Configuration file

After building the project, you can edit these settings with any text editor. Find the "remotingconfig.cfg" file in the "StreamingAssets" folder associated with your application. The configuration file uses standard JSON formatting.

If for some reason there is an error in the configuration file, and Hololight Stream can't parse it, Hololight Stream won't run correctly.

For more information on working with JSON, check out **json.org**.

2.5. Core components

2.5.1. Audio

Audio streaming allows you to stream the audio from your Hololight Stream Unity application to a connected Hololight Stream Client device. If your application has sound effects or music, audio streaming makes it possible to hear those on the client device. You can use the example Unity scene included with Hololight Stream to learn how to get started using Unity audio streaming yourself.

For more information about audio in Unity, check out Unity's page on the **Audio Source component**.

NOTE	The audio stream system only supports mono or stereo sound, 48 kHz sampling rate, and a DSP buffer size of 256, 512, or 1024 bits.

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices
- HTC devices
- Lenovo ThinkReality VRX[4]
- Magic Leap 2[4]
- Meta Quest devices
- Microsoft HoloLens 2
- PICO 4 Ultra

Example

The example in the Hololight Stream examples package uses a single audio source with basic functionality, but multiple audio sources with more complicated configurations also work.

1. Add the **Hololight Stream Examples** package into your Unity project using the **Package Manager** if you haven't already.
2. Using **Package Manager**, select **Hololight Stream Examples > Samples**.
3. Import the example, **Audio Stream Sample**.
4. Select **Hololight Stream Examples**.
5. Add an **Audio Listener** to the **Main Camera**, if the Main Camera doesn't already have one.
6. In the Project window navigate to **Assets > Samples > Hololight Stream Examples > 2025.0.0 > Audio Stream Sample**.
7. Add the **AudioStreamer.cs** component to the **Main Camera**.
8. Add the **AudioStreamExample** Prefab to the scene.

[4] Hololight Stream Client for Lenovo VRX and Magic Leap 2 is in maintenance. This means that these versions of Hololight Stream Client is either feature complete or deprecated.

9. Expand the **AudioStreamExample** GameObject and select the **StaticAudioSource** child GameObject.
10. In the **Inspector** go to the **AudioSource** component and add an AudioClip.

2.5.2. Camera

Camera stream is the sending of the client device's live camera feed to the server application along with some relevant metadata. Hololight Stream provides an API to control additional camera settings of the client device.

For more information on using the camera in Unity with AR Foundation, check out Unity's documentation on the **Camera**.

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices
- Microsoft HoloLens 2

Setup

Camera Settings

Setting	Description
Enable Camera	Enables camera stream capture and transmission.
Resolution and frame rate	Controls the resolution and frame rate of the camera stream.
Auto exposure	Turns on automatic exposure. This means you cannot use the slider to control the level of exposure for the camera. Exposure compensation still works whether auto exposure is on or off.

NOTE	On Microsoft Hololens 2, there is a white light on the front of the device that shows whether the camera is on or not.

Example

Open the **CameraStreamExample** scene or drag the **CameraStreamExample** Prefab into a scene. The example scene shows off the camera stream feature, rendering the camera stream onto a canvas and some tools for customization.

Camera Settings

XR Stream	When enabled, the 3D content in the application is rendered onto the real-world video feed from the device's camera.
Display metadata	When enabled, the received metadata is displayed in text format below.

2.5.3. Haptics

Haptics are a type of user feedback that create the sensation of touch through vibrations. Hololight Stream supports haptics through the Unity Input Subsystem.

Additional package requirements

- *None*

Supported client devices

- HTC Vive devices
- Lenovo ThinkReality VRX[5]
- Magic Leap 2[5]
- Meta Quest devices
- PICO 4 Ultra

[5] Hololight Stream Client for Lenovo VRX and Magic Leap 2 is in maintenance. This means that these versions of Hololight Stream Client is either feature complete or deprecated.

Set up

For information on how to set up haptics, check out:

- Unity's documentation on **HapticCapabilities**.
- Unity's documentation on **Simple Haptic Feedback** using the XR Interaction Toolkit.
- Microsoft's documentation on **IMixedRealityHapticFeedback Interface** in Mixed Reality Toolkit 2.

Buffered and non-buffered haptics

Using the input system, you can send non-buffered, or impulse, haptics and buffered haptics. Nonbuffered haptics are simple to send, allowing you to define frequency and amplitude. However, there is quite a bit of latency. Buffered haptics are faster and you can set far more complicated vibration patterns.

For more information, check out Unity's documentation on **InputDevice.SendHapticImpulse** and **InputDevice.SendHapticBuffer**.

Meta Haptics

Hololight Stream also supports ".haptic" files created by Meta Haptics Studio. You can import files exported from Meta Haptics Studio to Unity.
For more information on haptics and Meta Quest devices, check out Unity's article, **Haptics Overview**.

2.5.4. Microphone

Hololight Stream supports the remote capture of the client device's microphone.
An example is in "com.hololight.stream/Samples/Microphone/MicrophoneCaptureExample.cs". Add the **MicrophoneCaptureExample** prefab to the Scene or add the script to an existing GameObject in the Scene.

Additional package requirements

- *None*

Supported client devices

- HTC VIVE devices
- Apple iOS and iPadOS
- Lenovo VRX
- Magic Leap 2
- Meta Quest devices
- Microsoft HoloLens 2
- PICO 4 Ultra

Setup

To create and use the microphone stream, create an **IsarMicrophoneCapture** object and make sure to have an **AudioSource** attached to your gameObject. Add the **AudioClip** provided by calling the Start() method of IsarMicrophoneCapture and assign the AudioClip to the **AudioSource**. Additionally set the AudioSource to be looping and call the Play() method. To turn the Microphone off just call the Stop() function of IsarMicrophoneCapture. The example handles this behaviour directly when turning the object on or off.

NOTE	Don't forget to Dispose the IsarMicrophoneCapture object in OnDestroy().

2.6. Extensions

2.6.1. QR codes

Hololight Stream allows some client devices to read QR codes in the real-world environment and send that information back to the Hololight Stream application.

Hololight Stream only supports standard square QR Codes.

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices
- Microsoft HoloLens 2

Example

The ARQRCodeTrackerManager interfaces with the underlying subsystem to track QR codes.

AR QR Code Tracking Manager

The QR code tracking manager creates a GameObject for each QR code detected by the client device's camera. Once detected, you can read and use the information in the QR code. This manager can be found at the namespace "HoloLight.Isar.ARFoundation".

Respond QR code detection

Subscribe to the ARQrCodeTrackerManager's "trackedQrCodesChanged" event to be notified whenever a QR code is added (first detected), updated, or removed.

```
[SerializeField]
ARQrCodeTrackerManager m_QrCodeTrackerManager;

void OnEnable() => m_QrCodeTrackerManager.trackedQrCodesChanged +=
OnChanged;

void OnDisable() => m_QrCodeTrackerManager.trackedQrCodesChanged -=
OnChanged;

void OnChanged(ARQrCodeChangedEventArgs eventArgs)
{
      foreach (var newQrCode in eventArgs.added)
      {
            // Do something useful here on QR code added.
      }

      foreach (var updatedQrCode in eventArgs.updated)
      {
            // Do something useful here on QR code updated.
      }

      foreach (var removedQrCode in eventArgs.removed)
      {
            // Do something useful here on QR code removed.
      }
}
```

You can also get all the currently tracked QR codes with the ARQrCodeTrackerManager's "trackables" property. This acts like an IEnumerable collection, so you can use it in a foreach statement:

```
void ListAllQrCodes()
{
      Debug.Log(
            $"There are {m_QrCodeTrackingManager.trackables.count} QR
            codes being tracked.");

      foreach (var trackedQrCode in m_QrCodeTrackingManager.trackables)
      {
            Debug.Log($"QR Code: {trackedQrCode.trackableId} is at " +
                  $"{trackedQrCode.transform.position}");
      }
}
```

You can also get a specific image by its "TrackableId".

```
Or access a specific image by its TrackableId:

ARQrCode GetQrCodeAt(TrackableId trackableId)
{
      return m_QrCodeTrackingManager.trackables[trackableId];
}
```

Read QR code data

The "ARQrCode" class contains a byte array property called "data". This property contains the information read from the QR code. You can use this data to carry out specific logic based on the type of QR code.

The QR code must be UTF8.

QR code prefab

The ARQrCodeTrackingManager has a "QR Code Prefab" field. This is not intended for content. When a QR code is detected, ARFoundation will create a new GameObject to represent it.

If "Qr Code Prefab" is null, then AR Foundation creates a GameObject with an ARQrCode component on it. However, if you want every tracked QR code to also include additional components, you can provide a Prefab for ARFoundation to instantiate for each detected QR code. In other words, the purpose of the Prefab field is to extend the default behavior of tracked QR codes. It is not the way you should place content in the scene.

If you would like to instantiate content that matches the position and orientation of a detected QR code and have this update automatically, then you should set the content as a child of ARQrCode.

Enabling and disabling QR code tracking

The ARQrCodeTrackingManager has an "enabled" property, which you can set to enable or disable QR code tracking. When the application no longer needs to tracki QR codes, you should set the enabled property to false to improve performance.

More Info

For more information on using AR Foundation's trackable managers, check out Unity's AR **Foundation Trackable Managers page**.

2.6.2. Image tracking (2D)

2D image tracking is the detection and tracking of images in the real-world environment. You create a library of images for your application to recognize and Unity takes care of the rest. With AR Foundation, Hololight Stream supports this feature to varying degrees depending on the client device.

For more information, check out **Unity's AR Tracked Image Manager component page**.

NOTE	When running the Unity editor in Play mode, make sure to check the **Keep Texture at Runtime** checkbox for the images within the Image Library. If not, a warning within Unity will appear due to the texture not being available.

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices

2.6.3. Raycasts

You can use raycasts in AR Foundation to carry out hit tests on real-world objects.

The raycast feature only supports raycasts against planes.

For more information on using raycasts in Unity with AR Foundation, check out **Unity's AR Raycast Manager component page**.

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices

2.6.4. Spatial anchors

Spatial anchors are points in your virtual space that are aligned with the physical, tracked environment.

Additional package requirements

- Mixed Reality Toolkit 2 and Hololight Stream Extension for MRTK2
  or
- Mixed Reality Toolkit 3 and Hololight Stream Extension for MRTK3

Supported client devices

- Apple iOS and iPadOS devices
- Meta Quest devices
- Microsoft HoloLens 2

Set up

1. Open **Package Manager**.
2. Select **Hololight Stream Examples**.
3. Import the **Anchoring** sample.
4. Expand the **AR Session Origin** object and add the **AR Anchor Manager** to it.
5. Set **Anchor Prefab** on AR Anchor Manager. As example, a Spatial Anchor Object prefab has been provided.

Example

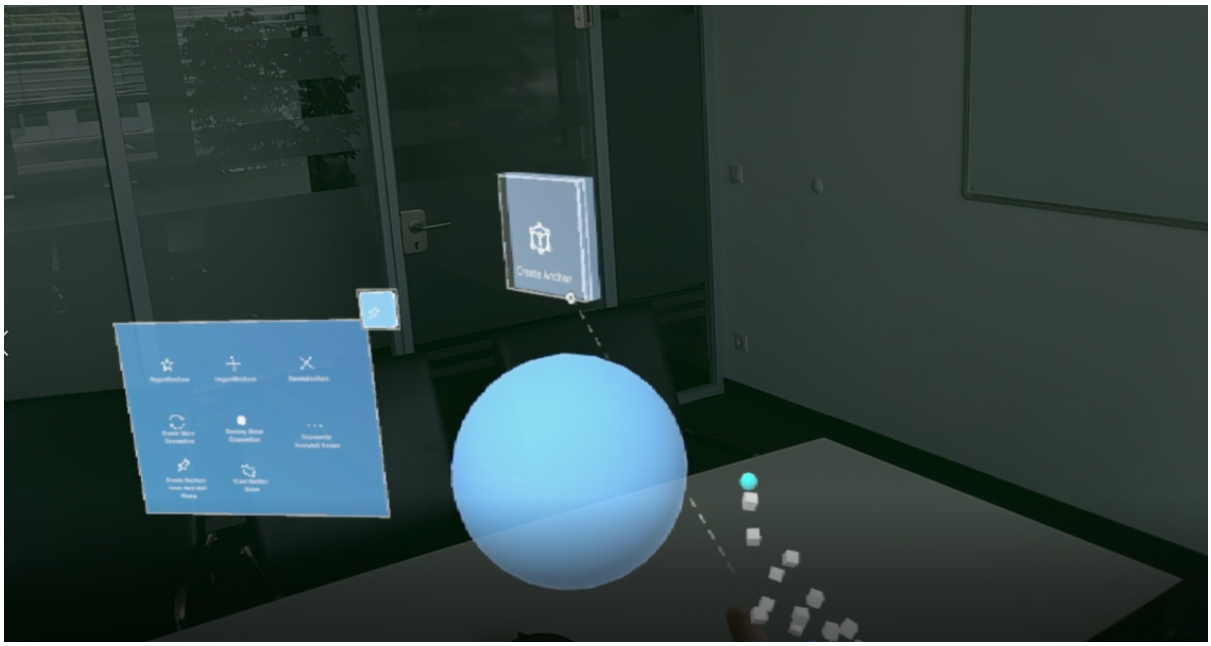
A pre-built scene is in the Anchoring sample for testing the anchors, To integrate the example into an existing scene, add the Anchor Example Prefab in the scene.

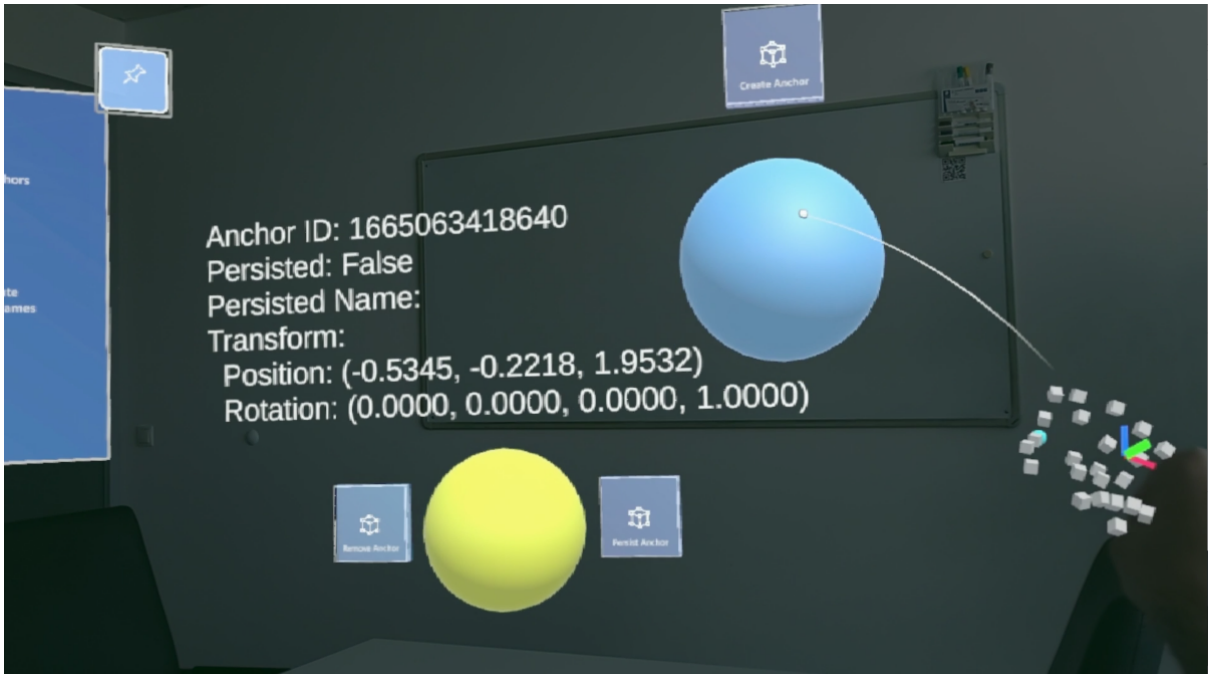
NOTE	The following steps use screenshots from the Mixed Reality Toolkit 2 example. There are visual differences between this MRTK2 implementation and the MRTK 3 example. Despite this, they both work the same.

Create and Delete spatial anchors

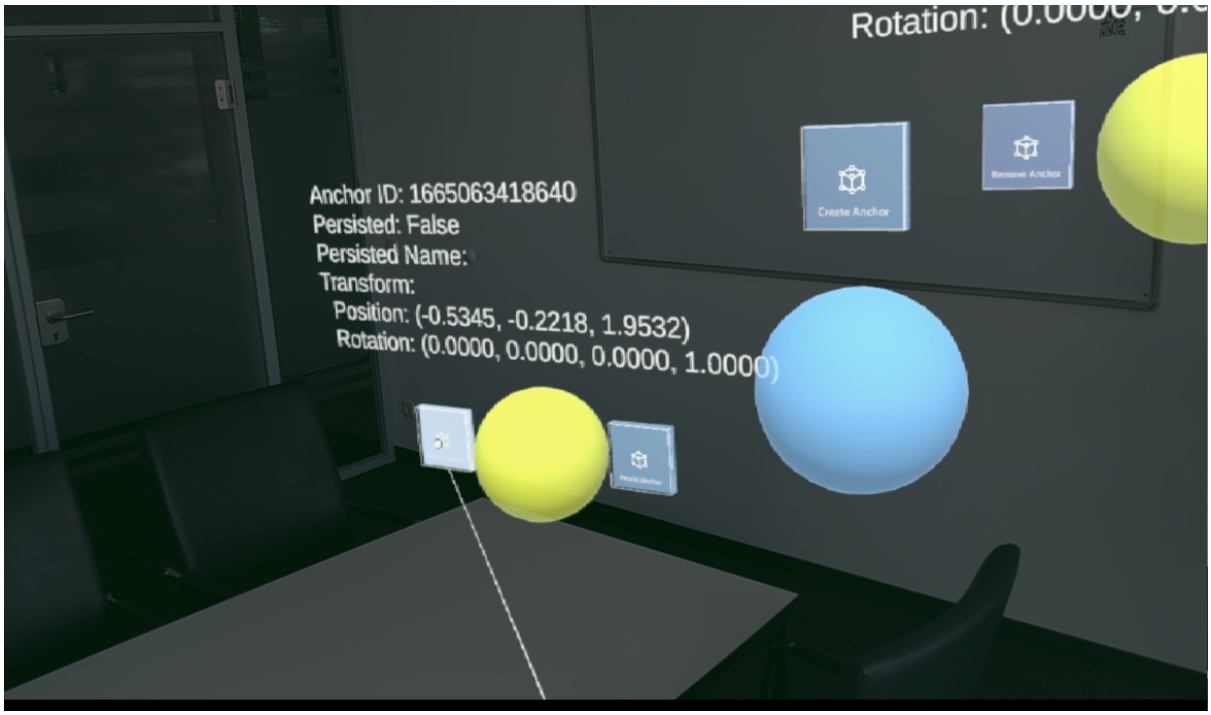
When running the example, you will see a control panel in the scene.

Create Anchor creates a spatial anchor at that position. The client device saves the original location and orientation of the anchors you create, updating their location and orientation every so often.





Any spatial anchor objects you create show up in the user's surroundings as yellow transparent spheres that you can select. Once selected, **Remove Anchor** removes the anchor from the system. **Persist Anchor** persists the anchor on the client device.



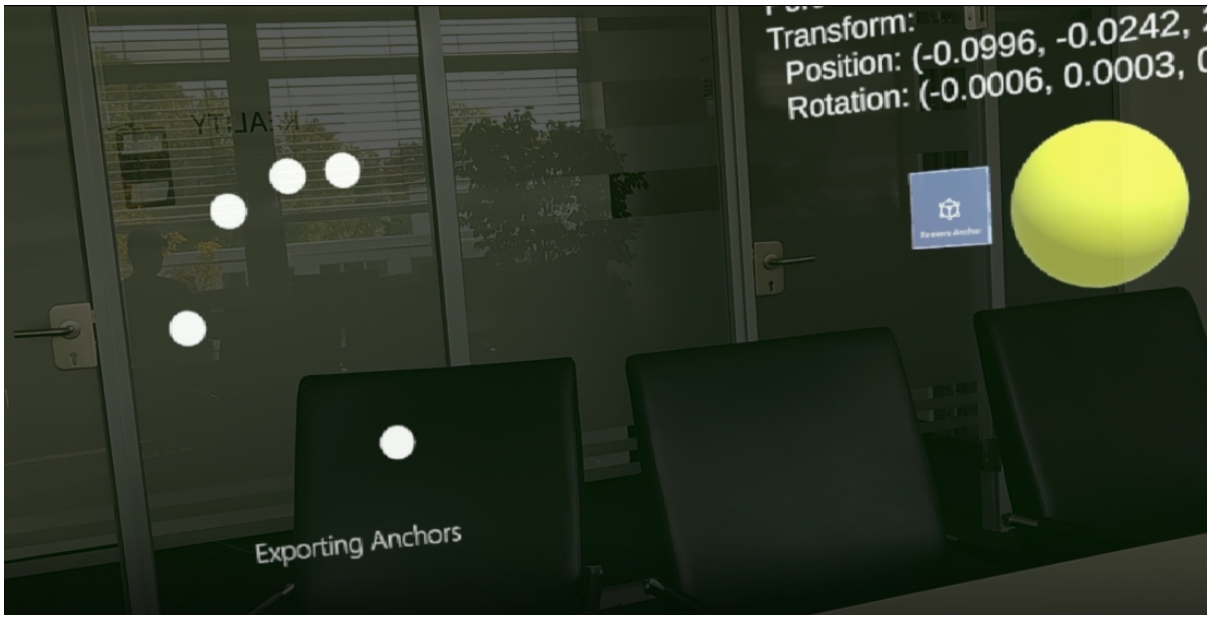
You can delete all spatial anchors by selecting **Delete Anchors** on the control panel.

Export and import spatial anchors

IMPORTANT	Exporting and importing spatial anchors is not supported on Meta Quest devices.

In the example, selecting **Export Anchors** in the control panel will download the file from the client and store it in Streaming Assets folder. Exporting spatial anchors can take a long time depending on the number of anchors. Because it can take so long the export function takes a timeout argument.

See “AnchorExampleUtils.cs” for more information about how to control the timeout logic.



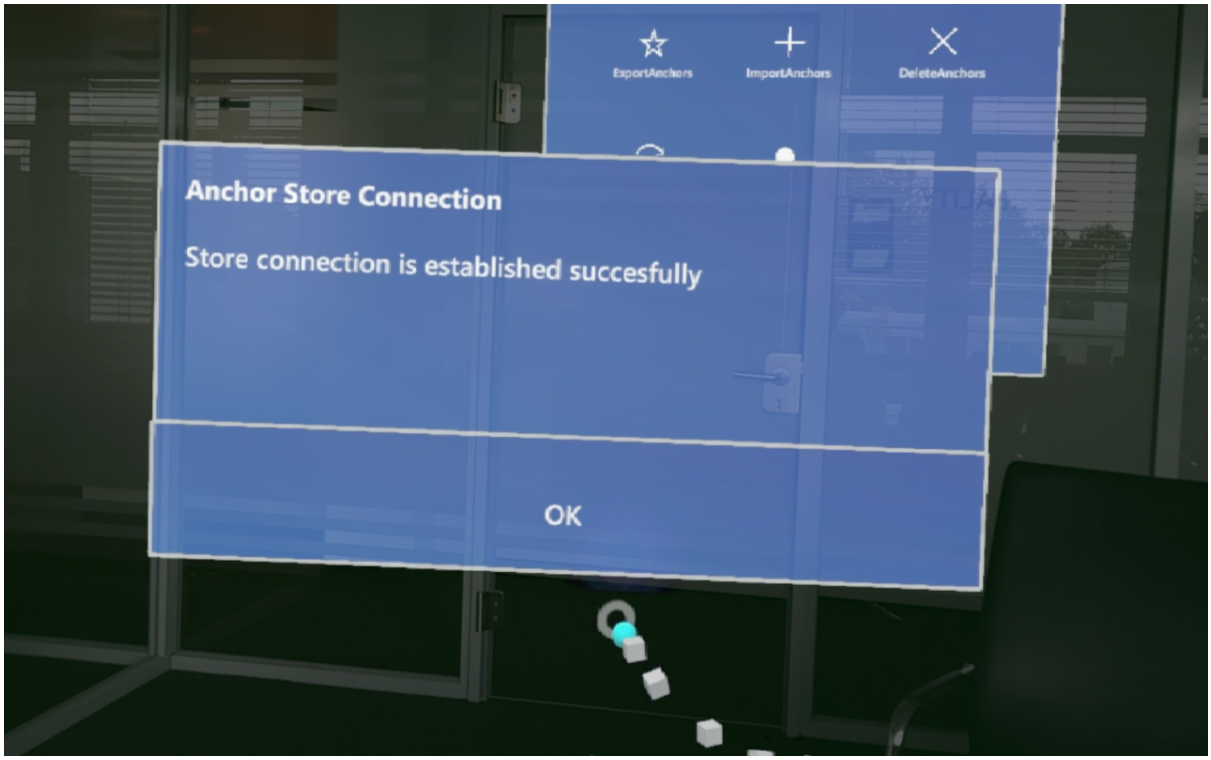
Once exported, you can import spatial anchor data onto the client device to load previously created spatial anchors back into the user's surroundings. In the example, selecting **Import Anchors** will upload the anchors data file from the Streaming Assets folder to the client device. Importing spatial anchor data can also take a long time, once again taking longer when there are more anchors. Because of this the import function also accepts a timeout argument.

See "AnchorExampleUtils.cs" for more information about how to control the timeout logic.

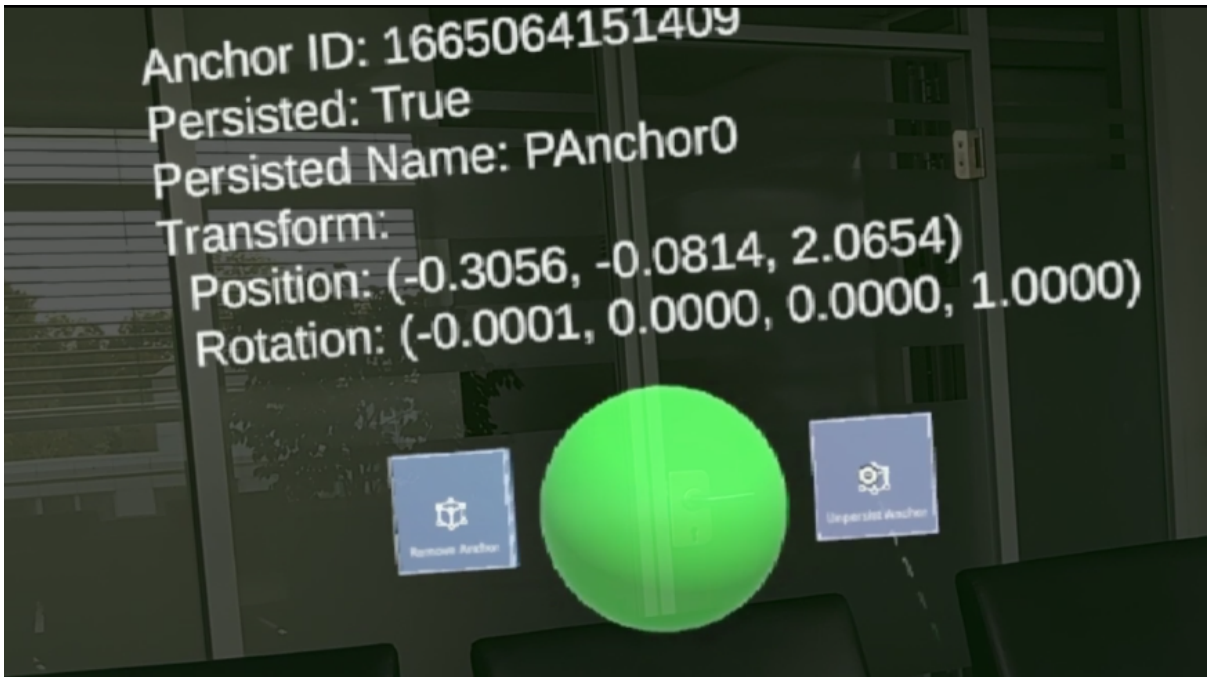
NOTE	Exporting or importing spatial anchors can fail due to issues in the network. Try to catch errors and try again.

Persisting anchors on the client device spatial anchor store

Instead of exporting and importing anchor data, you can also persist anchors directly on the client device. Persisted anchors are stored per application and exist on the device as long as the application is installed. To persist anchors you must establish a connection to the client's spatial anchor store. To do this, select **Create Store Connection** on the control panel.

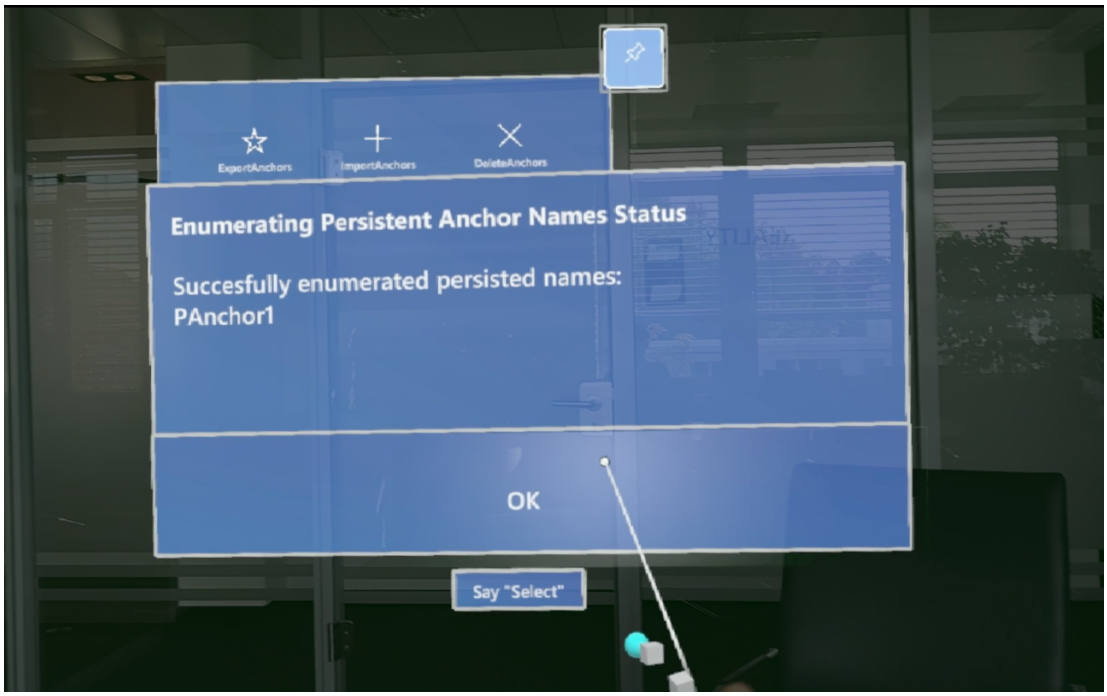


Each spatial anchor you create (represented by a transparent yellow sphere) has a **Persist Anchor** button. This tells the client device to persist the anchor to its anchor store. When selecting the button the color of the anchor will change form yellow to green and display additional persistence-related information. Additionally, **Unpersist Spatial Anchor** removes a persisted anchor from the anchor store.



The control panel has more ways to work with with spatial anchors:

- Enumerate persisted names
  Queries the persisted anchor names from the spatial anchor store on device.



- Create anchors from persisted names
  Loads all the persisted anchors from the anchor store of client device.
- Clear anchor store
  Clears all persisted anchor data from the client device.

The package also has an anchoring subsystem for adding, tracking, and removing real-world anchors.

For more information on how to use real-world anchors with AR Foundation, check out **Unity's Anchor Manager manual**.

2.6.5. Occlusion

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices

NOTE	This feature is currently only available on Apple devices equipped with a LIDAR sensor.

Detects whether Occlusion sends camera information for Apple iOS and iPadOS devices. Occlusion quality will increase depending on the device's sensor depth map resolution.

Put in the room mesh, it is possible to use it with HoloLens devices.

Depth information must be enabled to use this feature. To enable depth information:

1. Navigate to **Project Settings > XR Plug-in Management > Hololight Stream**.
2. Set the **Main Camera**'s alpha to 255 to achieve non transparent blending with the background.

MRTK2 can also display the room mesh and occlude.

To make things easier, there is a class called OcclusionExtension. This class manages the logic for toggling and configuring occlusion functionality on the client device.

Occlusion extension

The occlusion manager uses the underlying occlusion data channel. This data channel sends toggle and configuration messages to the client device.

The SendToggle() function toggles occlusion. "1" is on and "0" is off on the client device.

```
public void SendToggleWrapper()
{
	if (_occlusionExtension != null &&
	_occlusionExtension.IsConnected)
	{
		if (_isOcclusionEnabled == 0)
		{
			_isOcclusionEnabled = 1;
		} else
		{
			_isOcclusionEnabled = 0;
		}
		_occlusionExtension.SendToggle(_isOcclusionEnabled);
	}
}
```

The camera configuration can be sent (which includes near and far plane distance values of your camera) using the SendCameraInfo() function. This function requires a Camera GameObject to read camera information.

NOTE	SendCameraInfo() function must be called on Unity main thread, otherwise Main Camera will not be accessible.

```
public void SendCameraWrapper(Camera cam)
{
	if (_occlusionExtension != null &&
	_occlusionExtension.IsConnected)
	{
		_occlusionExtension.SendCameraInfo(cam);
	}
}
```

One important point is that you should subscribe to "OnConnectionChanged" event if you want to provide immediate occlusion information to your client device when connecting. You can achieve this using the method shown below:

```
private void OnEnable()
{
	_occlusionExtension =
	DataChannelManager.GetDataChannel<OcclusionExtension>();
	if (_occlusionExtension != null)
	{
		_occlusionExtension.Start();
		_occlusionExtension.OnConnectionChanged +=
OcclusionExtension_OnConnectionChanged;
	}
}

private void OcclusionExtension_OnConnectionChanged(bool connected)
{
	_readyToSend = connected;
}
private void Update()
{
	if (_readyToSend)
	{
	_occlusionExtension.SendCameraInfo(Camera.main);
	_occlusionExtension.SendToggle(_isOcclusionEnabled);
	_readyToSend = false;
	}
}
```

Example

For further information on how to use this feature, see the OcclusionController sample within the DataChannel folder of the "com.hololight.stream.examples" package. This sample sends configuration and toggle messages. This script can be added to an empty object in the scene to trigger the functions to enable or disable occlusion.

2.6.6. Object stabilization

Focus plane

A focus plane is used to stabilize content at a specific distance.

Additional package requirements

- *None*

Supported client devices

- Magic Leap 2
- Microsoft HoloLens 2

Set up

A focus plane, also known as the stabilization plane, is the focus point for the frame. The focus plane stabilizes holograms close to the focus point. You should set the focus point on every frame. You can do this using Unity's method, "XRDisplaySubsystem.SetFocusPlane".

To learn more about focus planes in Unity, check out **Unity's article about**

XRDisplaySubsystem.SetFocusPlane.

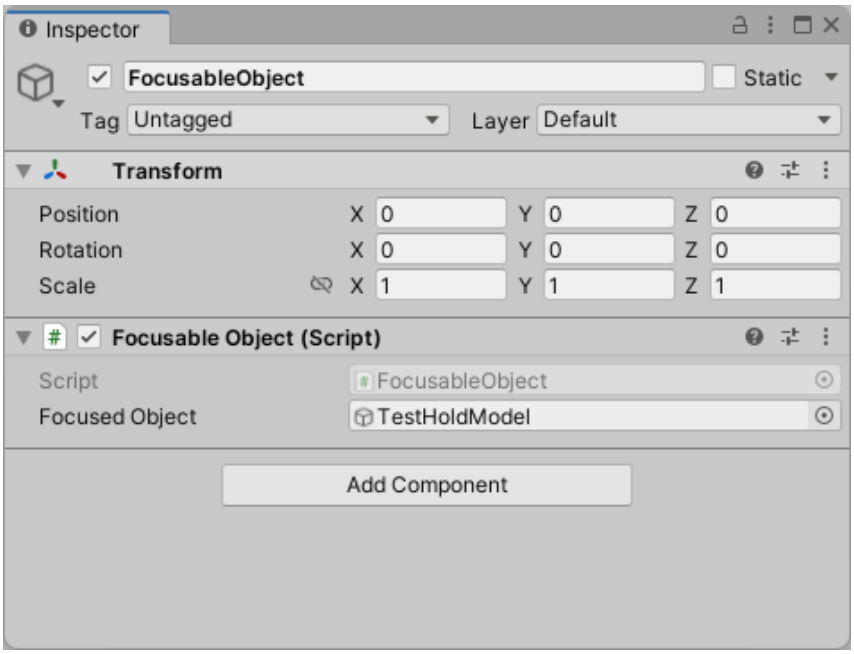
To learn more about focus point in MRTK, check out **Microsoft's article about the focus point in Unity**.

Before setting the focus plane, first you need to set "XRDisplaySubsystem.reprojectionMode" to "PositionAndOrientation". You can return this to "OrientationOnly" to disable focus plane stabilization when you no longer need it.

Example

An example on how to use focus planes is in the Hololight Stream examples package.

1. Add the **Hololight Stream Examples** package into your Unity project using the Package Manager is you haven't already.
2. Using Package Manager, select **Hololight Stream Examples > Samples**.
3. Import the example, **Object Stabilization Sample**.
4. After importing, search the Project and add the Prefab, **FocusableObject**, to your scene.
5. Select **FocusableObject** in the Hierarchy. In the Inspector under the Focusable Object component, you set **Focused Object** to the object you want focused.



Limitations

- Specifying a focus plane improves the stability of the object at the focus point, but decreases the stability of non-focused objects. It is up to you to decide which objects should be in focus and when.
- If a user passes through the plane they will experience undefined behavior. That means, you need to make sure that the user is never able to pass through the plane. You can prevent this either by moving the focus plane or disabling it entirely when the user gets too close.

Depth-based reprojection (preview)

Depth-based reprojection can improve the stability of all objects in the scene. While you are free to use this feature, remember that it is just a preview of the feature and may impact application performance when enabled.

Use the setting on Hololight Stream Client to enable depth-based reprojection.

Additional package requirements

- *None*

Supported client devices

- Microsoft HoloLens 2

Limitations

- Depth-based reprojection can negatively impact application performance and increase latency. This comes in the form of 3D objects having wobbly edges. This is much more apparent when one 3D object is in front of another relative to the user but the two 3D objects are still far apart from each other.
- Problems on the network will cause 3D objects to shake.

Head pose prediction

Head pose prediction (also called pose prediction) is the prediction of the future position and orientation of a user's head in order to create a smoother and more responsive XR streaming experience. This technique is crucial in combating the effects of latency.

Additional package requirements

- *None*

Supported client devices

- Lenovo Thinkreality VRX
- Meta Quest devices
- Magic Leap 2
- Microsoft HoloLens 2

Example

An example on how to use head pose prediction is in the Hololight Stream examples package.

1. Add the **Hololight Stream Examples** package into your Unity project using the **Package Manager** is you haven't already.
2. Using Package Manager, select **Hololight Stream Examples > Samples**.
3. Import the example, **Pose Prediction Configuration**.
4. After importing, search the Project and add the Prefab, **PosePrediction**, to your scene.

In the **Pose Prediction Config** component, the PosePrediction Prefab has an **Enabled** property with a checkbox that indicates whether pose prediction is on or not.

Under Configuration, you can select from multiple preset pose prediction configurations, including:

- Under Predict
- Moderate Predict
- Over Predict
- Custom

These presets change the two properties affecting the pose prediction algorithm according to how strongly you want pose prediction to work. If you select custom you can adjust the pose prediction properties yourself. These properties include:

- Prediction Tuner
  Adjusts the prediction level. Lowering this value can help reduce jitter.
- Prediction Cap
  Sets the maximum lookahead time in milliseconds for pose prediction. Lower values prevent the prediction when latency is high, which helps prevent jitter.

Limitations

- The streamed image may jitter or look unstable because of inaccuracies in the prediction model.
- Quick or sweeping movements can cause more prediction errors.

2.6.7. Scene understanding

Scene understanding detects and visualizes planes within the environment.

For more information on scene understanding in AR Foundation, known as plane detection, check out **Unity's AR Plane Manager component page**.

To use it as a part of MRTK's Scene Understanding feature, see **Spatial Awareness**.

Supported plane detection features by device

Example	Microsoft HoloLens 2	iOS	Meta Quest 2/Pro/3	Magic Leap 2
Arbitrary plane detection	Yes			
Boundary vertices[6]		Yes		
Classification	Yes	Yes		
Horizontal plane detection	Yes	Yes		
Data Channel Sample	Yes	Yes		

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices
- Microsoft HoloLens 2

[6] Hololight Stream Client for Microsoft HoloLens 2 does not support Boundary Vertices therefore, the boundary vertices correspond to the 4 vertices at the plane extents.

2.6.8. Meshing

AR Foundation provides meshing subsystem to receive and render the real world mesh. This can either be used directly with the subsystem, through AR Foundation's AR Mesh Manager or MRTK's Spatial Awareness.

Supported client devices

- Apple iOS and iPadOS
- Microsoft HoloLens 2

For more information on meshing in AR Foundation check out Unity's **AR Mesh Manager component page**.

Hololight Stream supports Unity's Meshing Subsystem to retrieve mesh data from the HoloLens 2 client. You can use this mesh data and display it using the MRTK Spatial Awareness System or the AR Foundations Mesh Manager.

Enabling MRTK Spatial Awareness

Additional package requirements

- Mixed Reality Toolkit 2

  or
- Mixed Reality Toolkit 3

Supported client devices

- Microsoft HoloLens 2

Set up

1. Open your Unity project.
2. In the scene, select **Mixed Reality Toolkit**.
3. In the Inspector, go to **Spatial Awareness > Enable Spatial Awareness System** and check the checkbox.
4. Set the **Spatial Awareness System Type** to **MixedRealitySpatialAwarenessSystem**.
5. Set the **profile** to **HololightStreamSpatialAwarenessSystemProfile**.

Additional Information

For more information regarding the how to edit the observer settings, check out Microsoft's page, **Spatial awareness getting started — MRTK2**.

NOTE	When using custom profiles for the Spatial Awareness system and the Spatial Awareness observer, it is important to create the correct profile types, as MRTK shows all profiles that use the same base profile type as possible candidates.

Performance considerations

The HololightStreamSpatialAwarenessSystemProfile by default uses the XR SDK Spatial Observer with its default settings. These settings can be edited in order to specify the functionality of the spatial observer. The following are performance considerations when editing the settings:

- Update Interval
  The update interval will determine how often the observer's center location is updated, in nonsatationary observer mode. When using a lower rate, the meshes will be updated more often and the application will be less performant.
- Observer Extents
  A larger bounding volume will display meshes in a larger area. If moving between rooms often, it is helpful to set a larger volume to avoid constantly adding and removing meshes from the space. When mostly in a single location, smaller volumes are often more performant.
- Level of Detail
  This will set the amount of triangles/cubic meter for the mesh generation. A finer mesh will reduce the performance of the application. Setting the value to custom will use the value set in Triangles/Cubic Meter. This is a value between 0.0-1.0, with 1.0 being the highest triangles/cubic meter.
- Display Option
  The display option determines how meshes should be displayed. The visible setting will generate display of meshes in the space, however it will be the least performant of the settings. When possible, disable this option and use the meshes for occlusion only.

Limitations

Unity currently only supports the Axis Aligned Cube bounding volume. Any other observer shape's will not be allowed. Recalculate Normals must be enabled to allow Unity to calculate the mesh normals. Without this, the generation will have unexpected behavior. If using a custom Level of Detail, the Unity will only accept a value between 0.0 - 1.0 for Triangles/Cubic Meter. Level of Detail setting is not applicable to iOS Client.

Enabling MRTK Scene Understanding

Additional package requirements

- Mixed Reality Toolkit 2

  or
- Mixed Reality Toolkit 3

Supported client devices

- Microsoft HoloLens 2

Set up

1. Open your Unity project.
2. In the scene, select **Mixed Reality Toolkit**.
3. In the Inspector, go to **Spatial Awareness > Enable Spatial Awareness System** and check the checkbox.
4. Set the **Spatial Awareness System Type** to **MixedRealitySpatialAwarenessSystem**.
5. Set the profile to HololightStreamSceneUnderstandingSystemProfile Additional information

For more information regarding the how to edit the scene understanding observer settings, see MRTK Scene Understanding Observer.

Hololight Stream uses its own Scene Understanding Observer instead of Windows Mixed Reality Scene Understanding Observer, so observer needs to be adjusted in the MRTK examples.

NOTE	When using custom profiles for the Spatial Awareness system and the Scene Understanding observer, it is important to create the correct profile types. This is because MRTK shows all profiles that use the same base profile type as possible candidates. "HololightStreamSceneUnderstandingObserver" uses its own custom profile type, so it needs to be checked carefully.

Performance considerations

The **HololightStreamSceneUnderstandingSystemProfile** uses the Hololight Stream Scene Understanding Observer by default with its default settings. You can edit these settings in order to specify the functionality of the spatial observer. The following are performance considerations when editing the settings:

- Update Interval
  The update interval will determine how often the observer's center location is updated in nonsatationary observer mode. When using a lower rate, the meshes will update more often and the application will be less performant.
- Query Radius
  A larger radius will display meshes in a larger area. If moving between rooms often, it is beneficial to set a larger radius to avoid constantly adding and removing meshes from the space. If remaining in a single location, a smaller radius is often more performant.
- World Mesh Level of Detail
  This will set the amount of triangles/cubic meter for the mesh generation. A finer mesh will reduce the performance of the application.

Limitations

- "World Mesh Level of Detail" does not support custom values.
- "World Mesh Level of Detail" setting is not applicable to iOS Client.

2.6.9. Gaze

Eye gaze uses the device tracking of the user's eye movements for input.

Gaze with MRTK2

Additional package requirements

- Mixed Reality Toolkit 2

Supported client devices

- Magic Leap 2
- Microsoft HoloLens 2

Set up

To make use of gaze interaction with MRTK 2, install and configure MRTK2 with Hololight Stream in your Unity project. Make sure to follow the instructions related to gaze functionality.

For more information on using gaze with MRTK2, check out Microsoft's article, **Gaze — MRTK2**.

For more information on using eye tracking with MRTK2 specifically, check out Microsoft's article, **Eye tracking in Mixed Reality Toolkit — MRTK2**.

Gaze with MRTK3

Additional package requirements

- Mixed Reality Toolkit 3

Supported client devices

- Magic Leap 2
- Microsoft HoloLens 2

Set up

To use gaze interaction in MRTK 3, install and configure MRTK2 with Hololight Stream in your Unity project. Make sure to follow the instructions related to gaze functionality. If a custom rig is required, create a Gaze Controller using the input actions provided in the Hololight Stream Extension for MRTK3. Make sure you add the Eye Tracking Manager, which is responsible for starting and stopping tracking, to the rig or to the scene.

For more information on using eye tracking with MRTK2 specifically, check out Microsoft's article, **Eye tracking — MRTK3**.

Gaze with Unity Input System

Eye tracking is also available in Unity's Input System. An example of this integration is available in the Hololight Stream example package.

Additional package requirements

- *None*

Supported client devices

- Magic Leap 2
- Microsoft HoloLens 2

2.6.10. Touch

Hololight Stream only supports single touch inputs on iOS and iPadOS clients.

Touch data is passed to Unity via the new Input System. This data can be accessed through actions which are configured with Unity Input Actions.

NOTE	Hololight Stream does not support any kind of screen space UI.

For more information on Unity's input system, check out their Input System **Quick start guide**.

Additional package requirements

- *None*

Supported client devices

- Apple iOS and iPadOS devices

Troubleshooting

If the touch inputs are not triggering events when running your application in the Unity Editor, make sure that the Unity Game window is actively selected. If it isn't, the events will not trigger.

2.6.11. Speech Recognition

Speech Recognition with Mixed Reality Toolkit

Speech Recognition allows you to interact with scene objects using a list of defined keywords. Saying these keywords in the client device's microphone will trigger the action.

Hololight Stream supports speech recognition for Microsoft HoloLens 2 when using the required Speech and Dictation Providers for MRTK. You can use dictation recognition input a continuous sentence to an object in the scene, for example an input field.

When using Hololight Stream Extension for MRTK2 or MRTK3, speech and dictation recognition are enabled default.

Additional package requirements

- Mixed Reality Toolkit 2 and Hololight Stream Extension for MRTK2
  or
  Mixed Reality Toolkit 3 and Hololight Stream Extension for MRTK3

Supported client devices

- Microsoft HoloLens 2

Configure the scene

To use Speech and Dictation, exchange the **Windows Speech Input** and **Windows Dictation Input** in the MRTK providers with **Isar Speech Input** and **Isar Dictation Input** respectively.

Example

An example on how to use Speech and Dictation can be found in the MRTK Examples Package. Guides on the respective examples can be followed and then migrated into Hololight Stream by applying the steps above.

After following the MRTK Extension Scene Configuration Step, the MRTK configuration profile will be overwritten, changing the SpeechCommandsProfile. Before running the example, set the **Input > Speech configuration** to **Speech.MixedRealitySpeechCommandsProfile**.

Speech Recognition without Mixed Reality Toolkit

You can also use speech recognition features without MRTK. To do this, use the IsarKeywordRecognizer and IsarDictationRecognizer classes.

Additional package requirements

- *None*

Supported client devices

- Microsoft HoloLens 2

Keyword recognition

"IsarKeywordRecognizer.cs" is responsible for keyword recognition.

Dictation Recognition

"IsarDictationRecognizer.cs" is responsible for dictation recognition.

NOTE	To use both recognition types at the same time, you should make sure to stop one before starting another. As the keyword recognition is done continuously and the dictation recognition is done with a timeout, manual stopping is only needed for keyword recognition. To check the status of the recognizers, Status property can be used. Default status is the stopped recognition. The Hololight Stream examples package includes an example on how to use this class.

2.6.12. Video passthough

Hololight Stream supports passthrough XR on the devices listed below.

In passthrough mode, every black object in the scene with an alpha value of 0 will not render. This means that objects or skyboxes with an alpha of 0 will not be displayed and instead replaced with the user's surroundings.

Refer to Unity's documentation for setting up different render pipelines.

Additional package requirements

- *None*

Supported client devices

- Apple Vision Pro
- Apple iOS and iPadOS
- HTC VIVE devices
- Lenovo ThinkReality VRX[7]
- PICO 4 Ultra
- Meta Quest devices

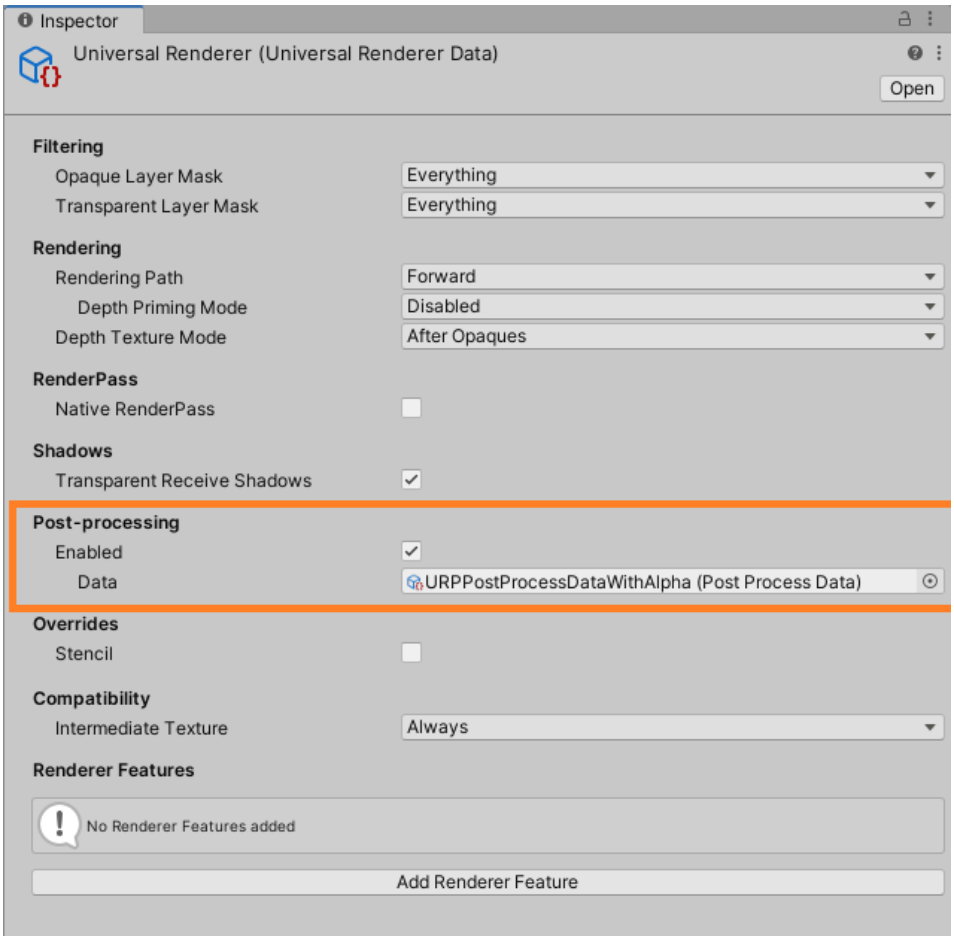
Alternate render pipelines

For different render pipelines check the Unity guides. Hololight Stream requires the background color to be solid black with an alpha value of 0.

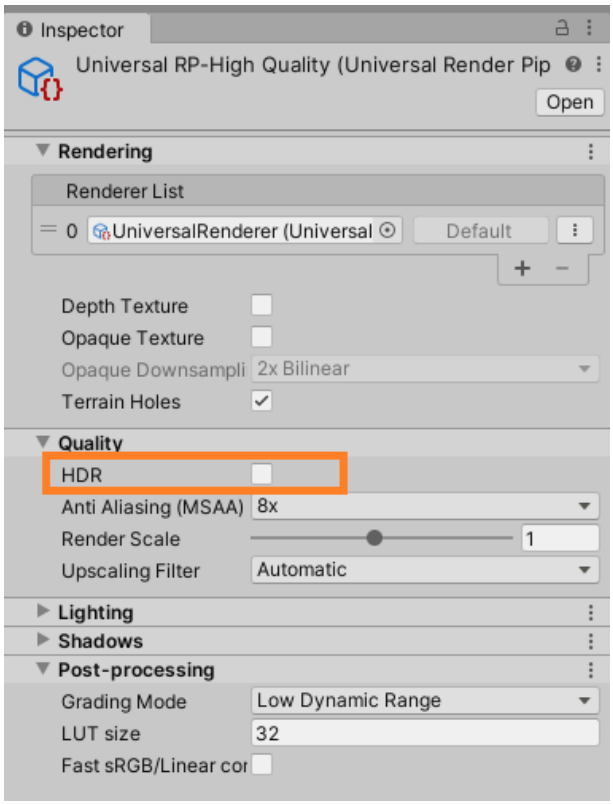
[7] Hololight Stream Client for Lenovo VRX and Magic Leap 2 is in maintenance. This means that these versions of Hololight Stream Client is either feature complete or deprecated.

Example

1. Create a 3D URP Unity project (the rest of this section has been done with the 3D Sample Scene (URP) available in the Unity Hub).
2. Follow the instructions in Hololight Stream examples setup.
3. 3. Open **Package Manager**. Import the **URPAlphaPassthrough** sample from the Hololight Stream Examples package.
4. 4. Go to **Assets > Settings > Universal Renderer (Univeral Renderer Data) > Post-Processing > Enabled**. Set **Data** to **URPPostProcessDataWithAlpha (Post Process Data)**.



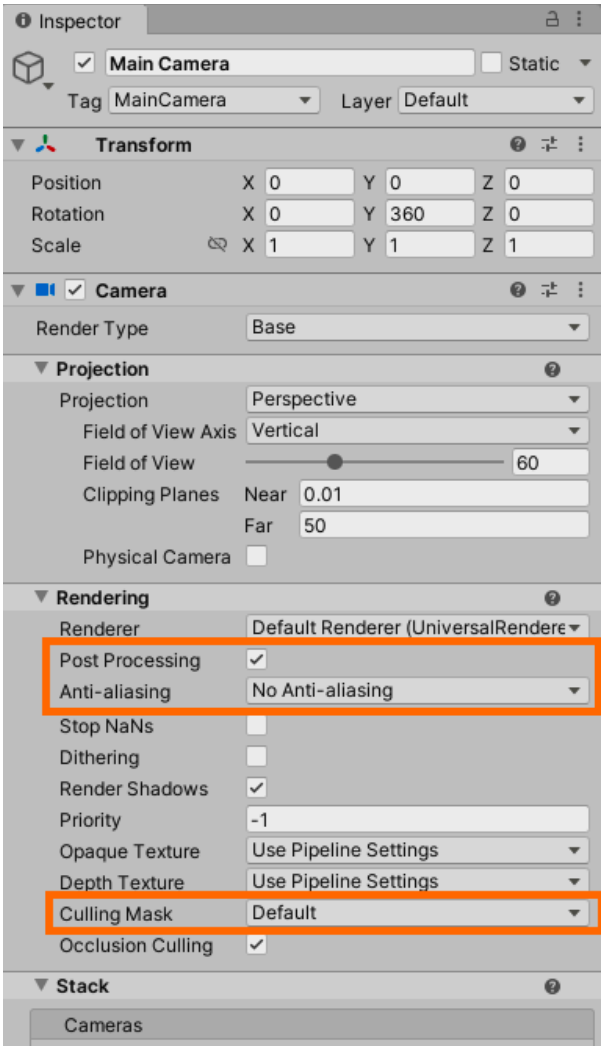
5. Go to **Assets > Settings > UniversalRP-High Quality > Quality**. Uncheck the box next to **HDR**.



6. Right click the Scene section and select **XR > Convert Main Camera to XR Rig** to use the scene with an XR device.
7. Go to **XRRig > Camera Offset > Main Camera** and in the Rendering sectio, set:

Camera Settings

Property	Value
Post Processing	*Box checked*
Anti-aliasing	No Anti-aliasing
Culling Mask	Default



8. Go to **XR Rig > Camera Offset > Main Camera** and, in the Background section, set:
   - Background Type to Solid Color
   - Background to zero in R, G, B and A
9. Enter **Play** mode and connect with Hololight Stream Client.

High Definition Render Pipeline

By default, the High Definition Render Pipeline (HDRP) does not render alpha information.

For more information, check out Unity's documentation, **Alpha channel configuration**.

To enable alpha rendering, follow Unity's steps below:

1. Open the **HDRP Asset** in the Inspector.
2. Go to **Rendering > Color Buffer Format**.
3. Select **R16G16B16A16**.

Troubleshooting

If you are having trouble getting passthrough to work, check the following:

1. Check that you enabled the passthrough setting on the client device.
2. Check that both the client and server are using the same color space.
3. Check that the Main Camera's background is a solid color with alpha value of 0.

Limitations

- Hololight Stream does not support transparency. Transparent objects will appear opaque. The passthrough image won't appear through them.
- To make the background transparent, make it black with an alpha value of 0.

2.7. Custom extensions

Custom extensions allow you to add features to Hololight Stream, allowing you to add new ways for the client and server to share information. You do this by creating data channels for your extension to use.

When you create a custom data channel, you need to make sure that you implement the new channel on both the server and client applications. This means you will need to create your own customized client application to make use of your new data channel.

If you want to create a custom extension, contact Hololight for more information.

2.7.1. Signaling

In order to establish a connection between two remote peers, they need to first exchange connection information. This process is called signaling. For more information, check out this **page on WebRTC signaling**.

Developers can provide their own signaling implementation to use in Hololight Stream in order to establish a connection with a remote peer. You might require custom signaling when using advanced connectivity functionality, complicated network environments, or when there are specific demands like security or encryption to exchange specific information before establishing a proper streaming connection. In most other cases going with the default implementation should be fine. There is a example implementation in the "com.hololight.stream.examples/Samples/Signaling" directory". If there isn't any signaling implementation provided, Hololight Stream will use the internal signaling implementation.

Signaling base class

A custom signaling implementation must inherit from the abstract base class "Signaling.cs".

Method	Description
Start	Called when the signaling process starts.
Stop	Called when the signaling process stops. This happens either when a connection is established or when the application has been closed.
ConnectionChanged	Called when the Hololight Stream connection state changes. Using this callback is at the discretion of the implementor.
SendSdp	Call when the SDP string is sent to the remote end.
SendIceCandidate	Called when the ICE candidate information is sent to the remote end.

You can call the following by implementing the class:

Method	Description
SetConnectionState	Notifies Hololight Stream about the signaling connection state. When called with a true connection state, Hololight Stream will asynchronously call SendSdp and SendIceCandidate to pass information to the remote endpoint. In situations where the implementor wants to carry out additional information exchange prior to Hololight Stream's information, this call can be delayed until the exchange is complete.
SetRemoteSdp	Notifies Hololight Stream about the obtained remote SDP.
SetRemoteIceCandidate	Notifies Hololight Stream about an obtained remote ICE candidate.

In the **XR Project Settings**, you can specify the signaling implementation Hololight Stream uses.

The implementation must be passed with the full path, including the namespace, to allow construction via System.Reflection.

The implementation is constructed and owned by the Hololight Stream XR Loader, IsarXRLoader class. A property, Signaling, is exposed by the loader which exposes the constructed object. This can be used to directly interface with the implementation specified within the settings file. For more information regarding the XR Loader and how to access it, see XRManagerSettings.

Example Unity server signaling

This example uses a simple TCP server which listens to a client on any local IP address. It sends the Hololight Stream information in XML format and prepends each message with the length of the XML string in big-endian format. This implementation can only be used with the default client signaling implementations provided.

NOTE	To specify the Hololight Stream Sample Signaling, enter "Hololight.Stream.Example.SampleServerSignaling, Hololight.Stream.Examples.Signaling" in the Signaling Implementation Type box.

The source code of this example can be found at “com.hololight.stream/Runtime/Signaling/SampleServerSignaling.cs”.

3. Hololight Stream Client

Hololight Stream Client runs on an array of supported devices, allowing them to connect to applications using Hololight Stream.

3.1. Supported Hololight Stream Client devices

Hololight Stream Client is available on the devices listed below. On many, but not all platforms, you can get Hololight Stream Client on a device's respective store.

Many, but not all, versions of Hololight Stream Client are also available at **Hololight Stream Downloads**.

As another option, you can go to the "Clients" folder in the Hololight Stream SDK and find the application files for the many of the supported devices.

3.1.1. Apple iOS and iPadOS

For information about the Apple iOS and iPadOS products, check out:

- **Apple iPad product page**
- **Apple iPhone product page**

Apple operating system compatibility

Device line	Supported operating system
iPad iPadOS 15.0 or newer	iPad iPadOS 15.0 or newer
iPhone iOS 15.0 or newer	iPhone iOS 15.0 or newer
iPod Touch	iOS 15.0 or newer

Install Hololight Stream Client on Apple iOS, iPadOS, and visionOS devices

Hololight Stream for iOS and iPadOS is only available from the Apple App Store.
Hololight Stream Client on the Apple App Store

3.1.2. Apple visionOS

For information about the Apple Vision products, check out:

- **Apple Vision Pro product page**

Apple Vision operating system compatibility

Device line	Supported operating system
Apple Vision visionOS 1.0 or newer	Apple Vision visionOS 1.0 or newer

Install Hololight Stream Client on Apple visionOS devices

Hololight Stream for visionOS is only available from the Apple App Store.
Hololight Stream Client on the Apple App Store

3.1.3. HTC VIVE Focus

HTC VIVE Focus device compatibility

For information about the HTC VIVE Focus 3 headset, check out the **HTC VIVE Focus product page**.

HTC VIVE Focus device	Hololight Stream supported
HTC VIVE Focus 3	Yes
HTC VIVE Focus Vision	Yes
HTC VIVE XR Elite	Yes

Install Hololight Stream Client on HTC VIVE Focus

Get the application package either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**.

For information on how to install applications onto the HTC VIVE Focus 3, check out HTC's instructions for **Installing APK files on the headset**.

3.1.4. Lenovo ThinkReality VRX

For information about the Lenovo ThinkReality VRX headset, check out the **Lenovo ThinkReality VRX product page**.

Install Hololight Stream Client on Lenovo ThinkReality VRX

Get the application package either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**.

For instructions on how to install Hololight Stream Client, check out **ThinkReality Cloud Portal – App Management**.

3.1.5. Magic Leap 2

For information about the Magic Leap 2, check out **the Magic Leap 2 product page**.

Install Hololight Stream Client on Lenovo ThinkReality VRX

Get the application package either from the "Clients" folder in the SDK or download it from **Hololight's download page**.

For information on how to install applications onto the Magic Leap 2, check out Magic Leap's 3 instructions on **Installing and Uninstalling Apps**.

3.1.6. Meta Quest

For information about the Meta Quest line of VR headsets, check out **Meta Quest product page**.
Quest-Stream-Client-2025.0.0.apk

Meta Quest device compatibility

Meta Quest device	Hololight Stream supported
Meta Quest 2	Yes
Meta Quest Pro	Yes
Meta Quest 3	Yes

Install Hololight Stream Client on Meta Quest

The easiest way to install the Hololight Stream client onto your Meta Quest device is using the Meta store. This also makes updates easier to install with the device as they become available.

You can also install Hololight Stream Client onto your Meta Quest device using an application package. Get the application package either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**. Install Meta Quest Developer Hub and connect it to your device. Follow the instructions on Meta's page **Get Started with Meta Quest Developer Hub**.

3.1.7. Microsoft HoloLens 2

For information about the Microsoft HoloLens 2, check out the **Microsoft HoloLens 2 product page**.

Install Hololight Stream Client on Microsoft HoloLens 2

The easiest way to get Hololight Stream Client onto your HoloLens 2 is by downloading and installing it from the Microsoft Store. This also makes updates easier to install directly with the device as they become available. You can also get the application package either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**.

3.1.8. PICO 4 Ultra

For information about the PICO 4 Ultra headsets, check out the **PICO 4 Ultra product page** and **PICO 4 Ultra Enterprise product page**.

PICO 4 device compatibility

PICO 4 device	Hololight Stream supported
PICO 4	No
PICO 4 Ultra	Yes
PICO 4 Ultra Enterprise	Yes

Install Hololight Stream Client on PICO 4 Ultra

Get the application package either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**.

For information on how to install applications onto PICO 4 Ultra and PICO 4 Ultra Enterprise, check out PICO's **Development resources**.

3.1.9. Web browser

You can set up a web server with Hololight Stream Client as a web application and use it in a web browser.

Supported web browsers

- **Google Chrome**
- **Microsoft Edge**
- **Safari (Apple)**

NOTE	Safari version 18.2 has an issue with audio decoding. When using Hololight Client on this version of Safari the application will crash whenever the connection starts. Versions 18.3 and 18.1.1 both work as expected.

Web browser Hololight Stream Client setup

Get the application either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**.

Host the published package as a web server with index.html in the root along with all the other files in the package. Navigating to the domain of the web application in your web browser will bring you to the client.

3.1.10. Windows

You can also use Hololight Stream Client on a PC running Windows.

Install Hololight Stream Client on Windows

Get the application package either from the "Clients" folder in the SDK or download it from the **Hololight Stream download page**. Extract all the files from "Hololight_Stream_Desktop_2025.0.0.zip" and run the executable.

3.2. Connect to Hololight Stream application with Hololight Stream Client

When an application using Hololight Stream is running correctly, it acts as a Hololight Stream server. This means you can connect to the application using a supported XR device and Hololight Stream Client. When you are connected you can use the application using Hololight Stream Client.

1. Open **Hololight Stream Client** on your supported device.
2. If not already selected, select the **Connect** tab on the left side.
3. Enter the **IP address** or **DNS** for the application in the field.
4. Select **Connect**.

After some loading you will connect to the application.

3.3. Hololight Stream Client settings

NOTE	Some settings are only available on some client devices.

3.3.1. Signaling port

Specifies the default port used for signaling. This port must be the same port number you specified as the server signaling port.

3.3.2. Timeout

The amount of time the client will try to establish a connection with the server. If this time passes and the client can't connect, the client will stop attempting to connect.

3.3.3. Bandwidth

Sets the maximum bandwidth limit to use during a connection. You can choose between the presets **Low**, **Medium**, and **High**. These presets describe the quality of the stream. You can also set a custom value.

3.3.4. Passthrough

You can turn passthrough on and off on supported devices. For virtual reality devices, turning on passthrough turns on the device's camera to switch the view to augmented reality. This means the user sees both the real world and virtual world together. For more information, see **Passthrough**.

3.3.5. Advanced settings

Preferred codec

Specifies the preferred codec. The preference will only be used if both the application and Hololight Stream Client support the codec type.

Color space

Specifies the color space which is used for rendering. This should be the same as what is specified in the server application.

Latency log

Enables logging of latency data. This data writes to a CSV file stored on the device running the client. You can use these logs to investigate networking behavior.

Each device has a specific place where Hololight Stream Client stores the latency log files. The name of the log file is always "Hololight Stream Log_<DATETIME>.csv".

Device	Log file path
Apple iOS and iPadOS	On My <DEVICE>\Stream Client
Apple Vision Pro	*Logging not available*
HTC VIVE	Internal shared storage\Android\media\com.holo_light.isar_client.config\log\latency
Lenovo ThinkReality VRX	Internal shared storage\Android\media\com.holo_light.isar_clien t.config\log\latency
Magic Leap 2	Internal shared storage\Android\media\com.HoloLightGmbH.ISA RClient\log\latency
Microsoft HoloLens 2	Folders\LocalAppData\Holo-LightGmBH.ISARClient_2025.0.0.0_arm64__<RAN DOM_STRING>\LocalState
Meta Quest	Internal shared storage\Android\media\com.holo_light.isar_clien t.config\log\latency
PICO 4 Ultra	Internal shared storage\Android\media\com.holo_light.isar_clien t.config\log\latency
Web browser	*Logging not available*
Windows	*Logging not available*