ClearVR SDK for Unity - Readme
v10.1.3 - 2023-08-22

Version information

Version: v10.1.3

Date: 2023-08-22

System requirements

- Unity: Unity 2018.4 (LTS) and up *)
- Windows: Windows 10 64 bit + Nvidia graphics card (Pascal and newer) or AMD graphics card
- Linux: recent Ubuntu, Debian or Arch-based 64 bit distro + Nvidia graphics card
- Android: Android v6.0 (Marshmallow, API level 23) and up, armeabi-v7a and arm64-v8a libraries are included
- iOS: iOS 11 and up
- Virtual machines, emulators and simulators are not supported **)
- A standards compliant hardware HEVC decoder is required
- A personal Tiledmedia license file is required to use this SDK.

*) The provided Unity Sample Player project requires Unity 2020.3.21f1+. **) x86 and x86_64 compatible binaries are provided for the Native SDK Android. While not officially supported, this will permit to build your app for the Android emulator.

ClearVR rationale

Streaming VR360 content over the internet at high quality proves to be virtually impossible with today's streaming technologies. Multiple approaches are being pursued to overcome this barrier. Tiledmedia's ClearVR approach relies on the fact that an ultra high resolution panoramic video is sliced-up in tiles. Only those tiles that are actually making up the viewport of the end-user are sent over the internet using standards compliant HTTP2. At any time, the entire panorama is streamed in low resolution as well making sure that there is always something to show when an end-user is quickly changing his viewport (e.g. rotating his head) and the system is still retrieving the correct tiles for that viewport. Our highly optimized network stack will typically retrieve the correct tiles from the network within one or a couple of frames, making sure that quality of experience greatly surpasses that of other solutions.

Our content is transformed prior to encoding and tiling and, after decoding the partially reconstructed panorama belonging to a specific field of view, tiles might not necessarily be at the location you would expect them to be (e.g. they are shuffled like a jigsaw puzzle). This means that all pixels on each video frame needs a specific unshuffling pattern to create a "normal" image again. A custom mesh and shader are used to achieve high performance frame accurate unshuffling.

More background information on ClearVR technology can be found here:
https://www.tiledmedia.com/index.php/technology/

The Unity Sample Player

This chapter gives a detailed overview on how to use the ClearVR SDK for Unity.

Introduction

Besides the ClearVR SDK for Unity (unitypackage), you have also received a UnitySamplePlayer ZIP package. This ZIP package contains a simple unity project that demonstrates how to interface with our SDK. Amongst other, it shows how to load, play, pause and seek in content, as well as performing fast channel switching (switching content), switching audio tracks, etc. This project is created in Unity `2020.3.0f1`. The Unity Sample PLayer project is set-up as follows:

1. A GameObject is placed at position (x=0, y=0, z=0) and rotation (x=0, y=0, z=0, w=1) in the scene.
2. `DemoPlayer.cs` is attached to this GameObject
3. A Camera is placed at position (x=0, y=0, z=0) and rotation (x=0, y=0, z=0, w=1) in the scene. This could be a normal Unity Camera or a Camera Prefab provided by a third party VR library like `OVRCameraRig` as found in the Oculus VR SDK.
4. A couple of buttons are included to demonstrate basic interaction with the player.

To build the project, make sure that:

1. Your test device has a working internet connection.
2. Your personal license file (`.tml`) has been added to Assets/Resources and has been renamed to `license.bytes`.
3. Your content list has been properly configured. This list can be found in Assets/Resources/content-list.json and includes two simple test clips by default.
4. The correct target platform is configured in Unity.

If running on supported OS and hardware, you can run the scene straight from the Unity Editor as well.

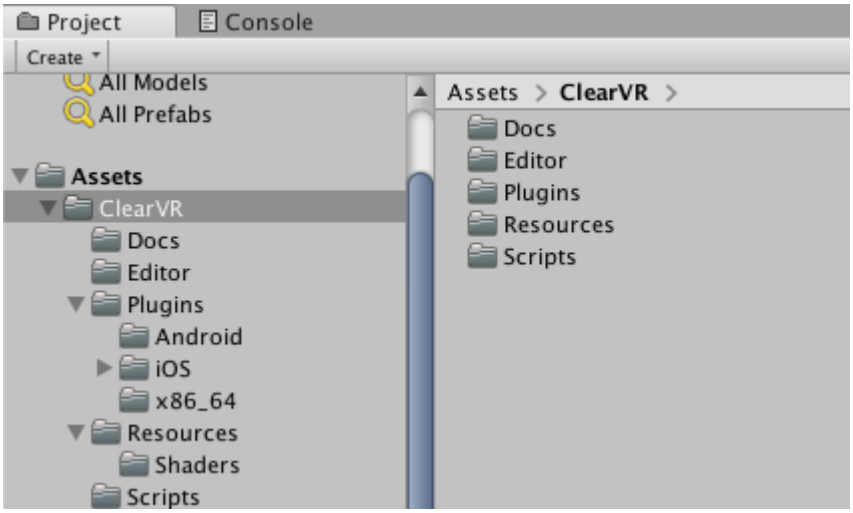
The ClearVR SDK has been designed to provide a platform agnostic interface to the integrator. This means that whether you are targeting iOS, Android or Windows, the APIs to interact with the player are identical and "behave" in the exact same way. As a general rule of thumb, feature parity can be expected across all platforms. There might be some minor differences when it comes to advanced features, features that are in beta or platform specific capabilities (e.g. Widevine DRM support is exclusive to the Android platform only, for obvious reasons).

Besides some helper classes and enum's, your primary interface will be the `ClearVRPlayer` class, as described in the next chapter.

ClearVR SDK Unitypackage

Introduction

The ClearVR SDK for Unity comes as a unity package. The contents of this package is shown below:



It consists of various scripts, shaders, a couple of editor scripts and plugins for the various supported platforms like Windows, Linux, Android and iOS. From time to time, you might receive updated to this unity package, typically asa drop-in replacement.

Updates

You might receive updates to the SDK from time to time. These updates are typically distributed as a drop-in replacement.

> **⚠ WARNING**
>
> The Unity Editor locks dynamic libraries (`dlls` on Windows, `.so` files on Linux) as soon as they are used for the first time. This might interfer with importing an updated unity package as it tries to replace those locked files. If you run into the problem, please restart the Unity Editor and re-import the unitypackage.

Graphics API support per platform

The ClearVR SDK supports various graphics APIs, depending on the target platform. The following table provides an overview.

	OPENGL CORE	OPENGLES 2	OPENGLES 3	METAL	DIRECT3D 11	DIRECT3D12	VULKA
Windows	v	-	-	x	v	x	x

	OPENGL CORE	OPENGLES 2	OPENGLES 3	METAL	DIRECT3D 11	DIRECT3D12	VULKA
Linux	v	-	-	x	x	x	x
Android	x	v	v	x	x	x	x
iOS	x	x	x	v	x	x	x
AppleTV	x	x	x	v	x	x	x

v = supported, x = not supported, - = might work (but not officially supported).

Notes:

1. Multithreaded rendering is supported on all platforms and graphics APIs.
2. OpenGLES support for iOS was dropped in ClearVR SDK v9.0. Only Metal is supported.

Unity XR plugin support

Since v9.0, the ClearVR SDK fully supports Unity's XR plugin management system in Unity 2019.3 and newer. It might work in older versions of Unity, but this is not guaranteed.

Codec support

The ClearVR SDK has been designed to provide optimal support for playback of spatio-temporally segmented high resolution content playback. Support for other formats are, however, being implemented. A list of supported formats:

1. ClearVR
2. HLS
3. (f)MP4 and progressive MP4

*) On all platforms, AVC and HEVC video codecs are supported as well as the AAC-LC audio codec.

Support for other formats and/or codecs to follow.

Content format support

CONTENTFORMAT	CODEC	DESCRIPTION
MonoscopicOmnidirectional	ClearVR	Monoscopic omnidirectional ClearVR content.

CONTENTFORMAT	CODEC	DESCRIPTION
StereocopicOmnidirectional	ClearVR	Sterescopic omnidirectional ClearVR content.
MonoscopicERP180	MP4, HLS	Monoscopic, 180 degree tranditional equirectangular content.
StereoscopicERP180SBS	MP4, HLS	Stereoscopic side-by-side, 180 degree tranditional equirectangular content.
MonoscopicERP360	MP4, HLS	Monoscopic, 360 degree tranditional equirectangular content.
StereoscopicERP360TB	MP4, HLS	Stereoscopic top-bottom, 360 degree tranditional equirectangular content.
MonoscopicFishEyeEquiSolid180	MP4, HLS	Monoscopic, 180 degree equisolid fisheye content.
StereoscopicFishEyeEquiSolid180SBS	MP4, HLS	Stereoscopic side-by-side, 180 degree equisolid fisheye content.
MonoscopicFishEyeEquidistant180	MP4, HLS	Monoscopic, 180 degree equidistant fisheye content.
StereoscopicFishEyeEquidistant180SBS	MP4, HLS	Stereoscopic side-by-side, 180 degree equidistant fisheye content.
MonoscopicRectilinear	MP4, HLS	Monoscopic, traditional broadcast content.
StereoscopicRectilinearTB	MP4, HLS	Stereoscopic top-bottom, traditional broadcast content.
Planar	ClearVR	Monoscopic, ultra-wide and/or ultra-high resolution "flat" content.

Note. MP4 = progressive MP4. See previous chapter for codec details.

Digital Rights Management (DRM)

The ClearVR SDK has support for Digital Rights Management (DRM) protected content playback. Support for other protection schemes (like Fairplay and PlayReady) are considered. Currently, AES-128 and Sample-AES are supported on all platforms for HLS content and Widevine (L3 and L1) is

supported on Android and iOS (but not tvOS).

Android

Due to fundamental limitations in Unity (all versions), Widevine L1 cannot be supported on Android in Unity. Widevine L3, however, is supported. As an exception, Widevine L1 encrypted content can be played on Oculus Android hardware like the Quest 1 and Quest 2 when rendering onto an OVROverlay. Refer to TextureBlitModes.OVROverlayZeroCopy for important details. When playing Widevine L1 protected content, one has to set `platformOptions.contentProtectionRobustnessLevel` to HWSecureAll.

Also note that OVROverlay based playback is only supported when using OVRPlugin v1.59.0 and newer.

Platform specific remarks

The following section contains platform-specific remarks.

Android

The following section is specific to the Android platform

AndroidManifest.xml

On Android, the ClearVR SDK relies on an*essential* bootstrap process in the `Activity`'s `onCreate()`. Without bootstrapping ClearVR in `onCreate()` you application *will* fail and experience random crashes. The ClearVR SDK comes with a couple of precompiled `Activity` that already contain the bootstrap process for regular android apps as well as various popular headset SDKs that are known to override the Android Activity themselves. You can use the ClearVR -> Android -> Setup Manifest XML menu in the Unity Editor to configure the applications Activity entry point (as configured in the `AndroidManifest.xml`) correctly. For convenience, you can refer to the following look-up table:

PLATFORM	ACTIVITY
No headset or Oculus	`com.tiledmedia.clearvrforunityandroid.ClearVRForUnityAndroidActivity`
WaveVR SDK	`com.tiledmedia.clearvrforunityandroid.ClearVRForUnityForWaveVRAndroidActivity`
PicoVR SDK	`com.tiledmedia.clearvrforunityandroid.ClearVRForUnityForPicoVRAndroidActivity`
PicoXR SDK	`com.tiledmedia.clearvrforunityandroid.ClearVRForUnityAndroidActivity`
Skyworth SDK	`com.tiledmedia.clearvrforunityandroid.ClearVRForUnityForSkyworthVRAndroidActivity`

⚠ WARNING

Newer Unity projects use templates instead (or adjacent to) a classic `Assets/Plugins/Android/AndroidManifest.xml`. For these project setups, the `Setup Manifest XML` menu entry does not cater. Be sure to set this up properly yourself instead!

If you happen to override the Activity yourself, you still*must* perform the ClearVR bootstrap procedure in your Activity's `onCreate()` as early-on as possible. Please refer to the following code snippet to perform conditional bootstrapping:

```
public class YourActivity extends UnityPlayerActivity {
    private static final String TAG = "ClearVRActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*
        To properly use the ClearVR library, one must bootstrap it in the main activity's onCreate().
        This will make sure that thread priorities and thread affinities are properly setup and that
        the ClearVRCore library is loaded into memory as soon as possible. This is **essential** as
        loading this library just in time WILL cause segfaults in Unity due to an obscure race-
        condition in Unity's garbage collector.

        The following debug message will be printed to logcat in case bootstrapping was completed
        successfully:

        "[ClearVR] Bootstrap completed successfully."

        Here, we use reflection to find our bootstrap method in the ClearVRCoreWrapper library.
        The customer has full flexibility on how he handles the Exceptions. Just remember that you
        cannot use ClearVR playback if bootstrapping fails.
        */
        try {
            Class<?> clearVRBootstrapClass =
Class.forName("com.tiledmedia.clearvrcorewrapper.ClearVRBootstrap");
            Method bootstrapClearVRMethod = clearVRBootstrapClass.getMethod("bootstrapClearVR");
            bootstrapClearVRMethod.invoke(null);
        } catch (InvocationTargetException argException) {
            throw new RuntimeException(String.format("Cannot load ClearVR libraries. Error: %s",
argException.getCause().toString()));
        } catch (Exception argException) {
            throw new RuntimeException(String.format("Cannot load ClearVR libraries. A generic error was thrown:
%s", argException.toString()));
        }
    }
}
```

If done correctly, you will see the following line in your logcat at the very beginning of the process's lifecycle (e.g. within the first 10 lines of the application-filtered logcat):

`D/[ClearVR]: Bootstrap completed successfully. Version: ...`

Oculus SDK compatibility

The ClearVR SDK is fully compatible with the Oculus SDK, and supports all headset on both the Android and Windows platform using the regular Oculus SDK and Unity integration. Multi-pass, single-pass and instanced stereoscopic rendering through optimized shaders. The GearVR And Oculus Go are still supported, but per the limitation as discussed here: https://developer.oculus.com/blog/gear-vr-app-development-no-longer-supported-from-sdk-suite-142-and-future-versions/

WaveVR SDK compatibility

The WaveVR SDK is supported. Stereoscopic playback, however, is currently not supported as the SDK does not properly support the `unity_StereoEyeIndex` shader variable. The ClearVR SDK will fall-back to monoscopic playback instead.

PicoVR SDK compatibility

The PicoVR SDK is fully supported. However, there is a known issue on Unity 2019.2.12 where you might experience random crashes when using the `mono` compiler. The `il2cpp` compiler does not suffer from this problem. Alternatively, one is suggested to upgrade to a newer version of Unity.

Skyworth SDK compatibility

There are no known issues and/or limitations w.r.t. Skyworth SDK support.

Cardboard and Daydream SDK compatibility

As a rule of thumb, ClearVR SDK support follows Unity's cardboard support. Daydream is no longer actively supported but is known to be working correctly on both Unity 2017 and Unity 2018.

iOS

iOS is supported from Unity 2018.3 and higher, but Unity 2020.3+ is highly recommended. The SDK comes with a PostProcess script that will configure the generated `xcodeproj`. This PostProcess script will make the following changes:

- Select the correct framework architecture based on target device (read device, tvOS or simulator).
- App target: add `MediaFlow.framework` as embedded framework but NOT add it to the Linked libraries sections.

In additon, for Unity 2019.3+:

- UnityFramework target: add `MediaFlow.framework` as framework (set to `Do not embed`) and add it to the Link binary with libraries section (as a Required library).

NOTE

Bitcode support was added in v8.0, but has been removed in v10.0 since Apple has deprecated support for it in XCode 14.

Windows

The Unity Editor for Windows as well as stand-alone builds are fully supported on recent (Pascal generation and newer) Nvidia graphics cards as well as recent AMD graphics cards.

Linux

Support for Linux is experimental and requires a recent Nvidia or AMD GPU.

The ClearVRPlayer class

The ClearVRPlayer is a `MonoBehaviour` object that manages the life-cycle of the player object. You can use its publicly exposed interfaces to load a clip, play, pause, seek, stop, switch to another clip, etc. Furthermore, it provides APIs to control how the video is rendered (e.g. force a stereoscopic clip to be shown as monoscopic) and it will provide you various callbacks (ClearVREvent) throughout its life. Interaction with the is typically asynchronous, meaning that your request (e.g. Seek +30 seconds in this current clip) will be met with a response at the point in time when the request was completed. The time between performing a request and receiving a response typically ranges from milliseconds to seconds, depending on the type of the request.

The ClearVRPlayer object itself has only a hand-full of public methods and fields. Interaction with the ClearVRPlayer is split across a number of interfaces which will be discussed in another section below.

How to use

The ClearVRPlayer class, being a `MonoBehaviour`, is added to a GameObject like any other: `gameObject.AddComponent<ClearVRPlayer>()`. Next, you should add an event listener using `clearVRPlayer.clearVREvents.AddListener(this.CbClearVREvent);`. This will ensure that your callback will be triggered on every event/state-change/message. Before you can Initialize() the ClearVRPlayer, however, you need to set-up the `PlatformOptions`. Please refer to the API documentation for details, but at minimum, you should set the following fields:

- `byte[] platformOptions.licenseFileBytes` holds your private license file data.
- `Transform platformOptions.trackingTransform` holds the transform of the currently active Camera, e.g. MainCamera or the CenterEyeAnchor on your OVRCameraRig.
- `ContentItem platformOptions.prepareContentParameters` holds the content item you want to play and the preferred intiial playback start position.
- `bool platformOptions.loopContent` specifies whether to loop content or not. Default value: true

You can keep all other fields to their default value. Next, you should call Initialize(PlatformOptionsBase argPlatformOptions, ...) to initialize your ClearVRPlayer object (*).

First, life-cycle of the ClearVRPlayer object and associated the `EventListener` will be discussed. Subsequently, the various interfaces exposed by the ClearVRPlayer will be covered.

(*) Please note that you *cannot* change any field on the platformOptions struct after calling `Initialize()`. Doing so is illegal and can result in undefined behaviour.

Life-cycle and the event listener

The ClearVRPlayer exposes an events channel to which the integrator should subscribe to. The

signature of the callback is as follows:
`void CbClearVREvent(ClearVRPlayer argClearVRPlayer, ClearVREvent argClearVREvent) { }`.

Please refer to the API documentation for exact details on the ClearVREvent struct, but in short:

The `ClearVREvent` has:

- a `ClearVREventType` field defining the type of the vent (e.g. `StateChangedBuffering`)
- a `ClearVRMessage` field that provides more details about the event.

This `ClearVRMessage` field is a struct in itself that has:

- a `ClearVRMessageTypes` field that can be either: `Info`, `Warning`, `FatalError`
- a `code` field that can have a value from the `ClearVRMessageCodes` or `ClearVRCoreErrorCodes` enums.
- a `message` string that optionally holds more information (e.g. in case of a `Warning` in can contain detailed information that warning).

The will notify you of any state change, event, error or warning through these ClearVREvents. A number of special `ClearVREventTypes` have been defined (like , while other events will be received as a generic .

There are a large number of ClearVREvents defined, the most important ones are listed in the following table:

CLEARVREVENTTYPE	DESCRIPTION
StateChangedPreparingContentForPlayout	The ClearVRPlayer is loading the selected clip.
StateChangedContentPreparedForPlayout	The content is buffered and prepared for playout.
StateChangedPlaying	In this state, the player stack is rendering video and (optionally) audio.
StateChangedBuffering	The player's internal buffers have depleted and buffering is required to resume playback. You will receive `StateChangedPlaying` when playback is resumed.
StateChangedPaused	Playback has paused. You will receive `StateChangedPlaying` when resuming playback.
StateChangedSeeking	Transient state while seeking. You will receive `StateChangedPlaying` when seeking has completed and playback has resumed.

CLEARVREVENTTYPE	DESCRIPTION
StateChangedSwitchingContent	Transient state while switching content. When done, you will receive the `ContentSwitched` event. Playback has only commenced when you have received the `StateChangedPlaying` event.
StateChangedFinished	The current clip has finished playback (e.g. the end of the clip was reached). In this state, no new frames are generated until you seek to a new position in the content.
StateChangedStopping	The ClearVRPlayer is in the process of destruction. It is essential for this to complete, so wait for StateChangedStopped until you Destroy() the ClearVRPlayer object!
StateChangedStopped	The ClearVRPlayer object is completely cleaned-up, has released all its resources and is ready for destruction. You will receive no further events after this state change.
ResumingPlaybackAfterApplicationPaused	Triggered when the player is resuming after the application has lost focus (e.g. was pushed to the background).
GenericMessage	A generic message can contain information information, a warning or a fatal error. You can check the embedded ClearVRMessage for details.

The `GenericMessage` can contain anything from harmless informational information to a `FatalError` that needs immediate attention. Query the `ClearVRMessage` embedded in the `ClearVREvent` to figure out what is going on.

> **TIP**
>
> It is recommended to hook up your visual buffering indicator to `StateChangedPlaying`, `StateChangedPaused` and `StateChangedBuffering` events.

The Interfaces

As explained previously, the ClearVRPlayer object exposes a few interfaces. Each interface in itself exposes various APIs that you can call. The following table briefly summarizes each exposed interface:

INTERFACE	DESCRIPTION	EXAMPLE API(S)
mediaPlayer	APIs related configuring the player for, and during, playback, but does not facilitate in controlling the playback itself.	ParseMediaInfo(), SetRenderMode()
controller	APIs related to controlling content playback.	Pause(), Unpause(), Seek(), SwitchContent()
mediaInfo	APIs related to acquiring (static) media info.	GetContentInfo()
performance	APIs related to performance statistics.	GetAverageBitrateInMbps()

Typically, you will be using the `controller` interface and its APIs most of the time.

Pausing and resuming playback

You can use Pause() to pause playback. Note that, when playing ClearVR content, pause might take a few video frames before it has completed. When Pausing, you will first receive a `StateChangedPausing` event before the `StateChangedPaused` ClearVREventType. The `StateChangedPausing` event is considered a transient state change and one is encouraged not to interfer. The provided `Action<>` will be triggered as soon as playback has paused.

Use Unpause() to unpause playback. You will receive the `StateChangedPlaying` event when playback has resumed. When playing ClearVR content, playback should resume right-away, but a short moment of buffering might be observed. The provided `Action<>` will be triggered as soon as playback has resumed.

Timing information

Various APIs are provided to query playback timing-related information. Please refer to the following table of definitions of timing-related concepts:

CONCEPT	DESCRIPTION
Content time	The absolute position in the clip since the beginning of the clip, with 0 (msec) defined as the very first frame in the video.
Content duration	The absolute length of the clip, defined as the difference between the content time of the very last frame and the content time of the very first frame (which is 0 by definition).
Wallclock time	The time relative to epoch (1-1-1970). Only applicable to LIVE content.

All timing is in milliseconds, unless noted otherwise.

One is strongly encouraged to use the GetTimingReport() API. This API returns a TimingReport object with information about the current position, lower and upper seek bounds. The unit of the values in this report depends on the specified `TimingType` (either in content-time or wallclock-time unit). This API is also useful in case of LIVE content, where the content availability is dictated by a continuously changing sliding window of segments available on the CDN.

> **⚠ WARNING**
>
> Do not call the GetTimingReport() API excessively. A couple of times per second should be more than sufficient.

Seeking

The ClearVRPlayer provides a powerful Seek() API. Seek behaviour depends on whether you are playing VOD or LIVE content as well as the SeekParameters that are used. Due to the nature of spatio-temporal video encoding, seeking will be performed up to the closest I-frame and may therefor not always be frame-accurate.

TimingParameters are set on the SeekParameters to specify the new target position in milliseconds and how this start position should be interpreted. Please refer to the following table on how the start position in interpreted:

TIMINGTYPE	INTERPRETATION (VOD)	INTERPRETATION (LIVE)
ContentTime	Absolute position	Absolute position
RelativeTime	Relative to current position	Relative to current position
LiveEdge	Undefined, do not use	Jump to the live edge
WallclockTime	Epoch position	Epoch position
Seamless	Ignored (see note)	Ignored (see note)
ScheduledOnDemand	Ignored (see note)	Ignored (see note)

Note that the provided position will always be bounded between 0 and the edge of the content. All times will be interpreted as milliseconds and "WallclockTime" must be provided in milliseconds since epoch. Typically, you would use the GetTimingReport for that value in conjunction with the SwitchContent API (see the next section).

If you want to quickly "jump forward" or "jump backwards", one should use the `RelativeTime` TimingType instead of querying the current content position and adding the preferred offset to it. The former works better when performing many seeks in rapid succession as the "current position" might not have changed on the outward-facing API yet by the time the next `Seek()` request is fired.

> **NOTE**
>
> TimingTypes.Seamless and TimingType.ScheduledOnDemand are not supported when seeking. Setting it results in undefined behaviour.

> **NOTE**
>
> The optional `SeekParameters.transitionType` field is not used yet. You can override the default value (`.Fast`), but the seek will always be executed as TransitionTypes.Fast.

Switching cameras / content

Fast content/camera switching is one of the core featurs of the ClearVR SDK. Assuming favorable network conditions, one can accurately switch between feeds within 40-80 milliseconds. This feature is exposed through the SwitchContent API.

`SwitchContentParameters` has a `contentItem` field to specify the target to switch too (see the ContentItem description below for details). Next, the TimingParameters can be tuned to configure the preferred start position in the content after the switch as well as how to switch should be performed. These are the same TimingTypes as described in the Seek section above. In case of SwitchContent, however, `Seamless` is a supported mode. Furthermore, the optional `transitionType` can be specified.

In case TimingTypes.Seamless is selected, the player stack wil attempt to perform a seamless switch to the next clip. In this case, the specified start position will be ignored. Best results are achieved if both sources are frame-accurately synchronized and under favourable network conditions.

The TimingTypes.ScheduledOnDemand mode can be used to signal that a VOD clip is scheduled to start at the specified wallclock moment in time. The `target` is interpreted as the wallclock time (in epoch milliseconds) at which the VOD asset should start playback. For example, if a VOD asset with a duration of an hour should be played back as a live event that starts on February 1, 2021 08h:00m:00s pm UTC, the target would be set to 1612987200000 (milliseconds). The playback behaviour of the `ClearVRPlayer` now depends on when the end-user joins the scheduled stream.

END-USER JOINS 8PM-9PM SCHEDULED STREAM	RESULT
before 8	Playback will be delayed until the `target` is reached

END-USER JOINS 8PM-9PM SCHEDULED STREAM	RESULT
between 8 and 9	Playback will start at the position in the stream relative to the configured `target`
after 9	The `StateChangedFinished` event will be emitted (or the clip will fail to load if the assets are no longer available on the CDN)

TIP

For `ScheduledOnDemand` seeking is not blocked by default, it is however currently not possible to seek relative to the scheduled start `target`.

Depending on the TransitionType, the content switch will be performed either `Fast` (default) or `Continuous`. A `Fast` switch will result in playback temporarily stopping while the switch is serviced as quickly as possible. A `Continuous` switch will result in current clip to continue playing until the next clip is sufficiently buffered for the actual switch.

NOTE

A `Seamless` timingType always implies a `Continuous` transitionType.

Android specific remarks

When playing DRM, one can freely switch to another clip that has the same or another encryption cypher. Furthermore, one can freely switch between Widevine L3 encrypted content and clear content. Switching from L3 to L1 encrypted content is not possible. Please refer to the DRM section above for further details on DRM. Also note that Widevine L1 support is severely limited because of Unity-specific limitations.

Sync

The ClearVR SDK supports synchronized playback of video streams across devices. This is currently supported on Android, iOS and PC.

For synchronized playback between multiple devices there needs to be concensus on a synchronization target, also known as sync edge. This sync edge is determined in different ways based on the content.

CONTENT TYPE	SYNC EDGE
Live	A distance behind the live edge calculated from the live stream properties, no client configuration necessary
Scheduled VOD	Based on the scheduled start time
Regular VOD	Not applicable

On enabling sync the player will initially seek as close as possible to the sync edge. As seek is limited to I-Frame positions, further adjustments will be made by smoothly increasing and decreasing the content playback speed until the sync edge is reached, with a precision in the order of a single frame. The algorithm uses reasonable defaults for playback speed manipulation, but more advanced control is possible by adjusting the parameters on the SyncSettings.

For supported content types and devices as specified above synchronized playback can by enabled by adding the SyncSettings to the `PrepareContentForPlayout` and `SwitchContent` calls, or while content playout is already active using EnableSync. Synchronized playback will be disabled on action impacting video playout, such as `SwitchContent` (without sync settings), `Seek` and `Pause`. It can also be explicitely disabled using DisableSync.

Stopping the ClearVRPlayer

This is an important topic that needs careful attention from the developer. The ClearVRPlayer needs a "little bit of time" (more than 1 VSync) to properly release all resources, therefore *you should never call* `UnityEngine.Destroy(clearVRPlayer);` *!* to stop it. Doing so can result in the ClearVRPlayer not releasing all its claimed resources as Unity has already destroyed our `MonoBehaviour` before it can wrap-up. Instead, one must call Stop() instead (see also the description of theClearVRPlayer interfaces in the next section) and wait for the `Action<>` callback to fire. In the callback, one can do the final clean-up (and only by that time one *is* allowed to `Destroy()` the clearVRPlayer).

Please note that the ClearVRPlayer can also decide for itself to stop, for example when it encounters a fatal error. Therefor, one *must* check for the `ClearVREventTypes.StateChangedStopped` `ClearVREvent` in his event listener. When you receive this event, you can assume that the ClearVRPlayer has released all resources and you can continue to doing your final clean-up like you would've done if you had received the callback on your `Stop(Action<>)` call as described in the previous paragraph. Note that, in case you call `Stop(Action<>)`, you will receive both the callback to your request as well as the `ClearVREventTypes.StateChangedStopped` event.

NOTE

When stopping playback in the Unity Editor (Windows, Linux), you might notice a brief "lag" between clicking the Unity "Play" button and the editor stopping. See the next section.

Application suspend/resume management

The ClearVRPlayer class can manage playback suspensions and resume when the application is pushed to the background (or the headset is put down) and when the application regains focus again. You can control the behaviour by adjusting applicationFocusAndPauseHandling. By default (Recommended) the underlying player will be destroyend upon app suspension and recreated from scratch upon app resume. This will ensure that all resources are freed and no additional battery drain is incurred while the app is suspended. Downside, however, is that playback will take a couple of seconds to resume after app resume. Alternatively, one can consider selecting Legacy. In this case, playback is paused (if not already paused) upon app suspend and resumed (if not paused before app suspend) upon app resume. We strongly advise against using this mode. If you want to disable any form of automatic player management upon app suspend/app resume, one can set this field's value to Disabled. Remember that with great power comes great responsibility. Tiledmedia cannot provide support on issues related to disabling player-related app-life-cycle management.

Other commonly-used APIs

The following table briefly discusses some of the commonly used APIs that have not been covered in the previous sections.

HOW DO I?	API
Check if current platform is supported	GetIsPlatformSupported
Get current position	GetTimingReport
Get content duration	GetTimingReport
Get current ContentFormat	GetContentFormat
Check if clip is LIVE or VOD	GetEventType
Temporarily switch to monoscopic rendering	SetRenderMode
Mute/unmute audio	SetMuteAudio
Control audio level	SetAudioGain

HOW DO I?	API
Get current average bitrate	GetAverageBitrateInMbps
Convert projection type to ContentFormat	ConvertMediaProjectionTypeToContentFormat
Know what my SDK version is	GetClearVRCoreVersion (*)
Know when ContentFormat has changed	Check the ContentFormatChanged event
Know when the first frame is rendered	Check the FirstFrameRendered event
Know if the clip is finished playing	Check the StateChangedFinished event

(*) Alternatively, check cover page of this document or `Assets/ClearVR/docs/version.txt`.

Telemetry support

v10 adds support for Telemetry reporting to backends of choice. Currently,NewRelic is supported. below is an example on how to configure a Telemetry Target on your platformOptions:

```
// Define telemetry target configuration (in this example for a NewRelic end-point)
TelemetryTargetConfigNewRelic telemetryTargetConfig = new
TelemetryTargetConfigNewRelic("YOUR_ACCOUNT_ID", "YOUR_LICENSE", "YOUR_END_POINT");
// Configure your telemetry target (you can have multiple)
TelemetryTarget telemetryTarget = new
TelemetryTarget(TelemetryIPSignallingTypes.TelemetryIpSignallingMasked, new
List<TelemetryTargetConfigBase>() { telemetryTargetConfig });
// Set the telemetry configuration.
platformOptions.telemetryConfiguration = new TelemetryConfiguration(new List<TelemetryTarget>() {
telemetryTarget });
```

Support for other telemetry target can be added on request.

The ContentItem class

The `ContentItem` class must be used when initializing the ClearVRPlayer, as well as when switching to a different clip. Use its `manifestUrl` field to specified the url. Specify the `TimingParameters` to configure the preffered playback start position. Refer to TimingParameters for details on how to configure it for VOD and Live clips. While ClearVR encoded content embeds the exact content format (e.g. 360 degree stereoscopic), Regular content like progressive MP4 and HLS do not (or if they do, they do so in an unreliable and barely standards-compliant way). You can use the `ContentFormat overrideContentFormat` field to specify the exact format of such a regular clip. Please refer to the `ContentFormat` enum description in the API documentation. Furthermore, a helper method is included in the SDK (`ContentFormat Utils.ConvertMediaProjectionTypeToContentFormat(String)`) that can assist in converting a string representation into a ContentFormat.

The optional `DRMInfo` can be used to configure DRM specific parameters. Please refer to the `DRMInfo` API documentation for details. Keep at `null` if your content is not DRM protected.

The `FishEyeSettings` can be tuned for regular content (e.g. progressive MP4 and HLS) that was recorded and encoded as a fish eye. A small number of pre-configured lens-types are included in the SDK. If your specific configuration is not supported out of the box, you can specify the parameters manually. be sure to set the exact correct parameters, being a bit off can result in totally skewed and/or broken video.

Note: Widevine DRM support is only available on the Android platform. Playback of ClearKey DRM, HLS AES-128 and Sample-AES (ClearVR and HLS) is available on all platforms.

The ContentInfo class

The ContentInfo class is used to describe (e.g. the resolution of available ABR representations, number of audio tracks, etc.) of a specific ContentItem. Each ContentInfo object contains N feeds as `FeedInfo` objects. these FeedInfo objects contain:

- An array of video tracks in the form of aVideoTrackInfo object
- An array of audio tracks in the form of aAudioTrackInfo object
- An array of subtitle tracks in the form of aSubtitleTrackInfo object The `VideoTrackInfo`, `AudioTrackInfo` & `SubtitleTrackInfo` each contain various fields to describe the track. Use `ContentInfo` in conjunction with the `ClearVREventTypes.ActiveTracksChanged` event to determine the active Feed(s) and Audio/Video/Subtitle track(s). Various helper methods are available on `ContentInfo` and `FeedInfo` to help in this process.

As stated, each `Feed` can have an array of `VideoTrackInfo` with each `VideoTrackInfo` representing a representation (e.g. ABR level). These `VideoTrackInfo` objects are guaranteed to be sorted as follows:

1. Higher resolution before lower resolution

2. In case of equal resolution: higher bitrate before lower bitrate.
3. In case of equal resolution and bitrate: HEVC before AVC.

Visualization of a `ContentInfo` object of a `ContentItem` with one Feed with two video track representations (e.g. two ABR flavors), one audio track and one subtitle track:

```
ContentInfo
 ├─Feedinfo
 │  ├─VideoTrackInfo
 │  │  ├─index
 │  │  ├─width
 │  │  ├─height
 │  │  ├─framerateNum
 │  │  ├─framerateDenom
 │  │  ├─contentFormat
 │  │  └─drmType
 │  ├─VideoTrackInfo
 │  │  ├─index
 │  │  ├─width
 │  │  ├─height
 │  │  ├─framerateNum
 │  │  ├─framerateDenom
 │  │  ├─contentFormat
 │  │  └─drmType
 │  ├─AudioTrackInfo
 │  │  ├─index
 │  │  ├─boundToVideoTrackIndex
 │  │  ├─codec
 │  │  ├─sampleRate
 │  │  ├─numberOfChannels
 │  │  ├─name
 │  │  ├─language
 │  │  └─drmType
 │  └─SubtitleTrackInfo
 │     ├─index
 │     └─boundToVideoTrackIndex
```

Visualization of a `ContentInfo` object of a mosaic `ContentItem` with two feeds (e.g. cameras) of which each feed contains two video track representations (e.g. two ABR flavors), one audio track and one subtitle track.


ContentInfo
Feedinfo
VideoTrackInfo
index
width
height
framerateNum
framerateDenom
contentFormat
drmType
VideoTrackInfo
index
width
height
framerateNum
framerateDenom
contentFormat
drmType
AudioTrackInfo
index
boundToVideoTrackIndex
codec
sampleRate
numberOfChannels
name
language
drmType
SubtitleTrackInfo
index
boundToVideoTrackIndex
FeedInfo
VideoTrackInfo
index
width
height
framerateNum
framerateDenom
contentFormat
drmType
AudioTrackInfo
index
boundToVideoTrackIndex
codec
sampleRate
numberOfChannels
name
language
drmType
SubtitleTrackInfo
index
boundToVideoTrackIndex

Display Objects and the Layout Manager

This section describes ClearVR's Display Objects, the Layout Manager and the concept of Display Object Mappings that were introduced in v9.0. Customer upgrading from previous versions are highly encouraged to start using these concepts. All v9.x releases contain a backwards compatible "Legacy" mode where the SDK tries to mimick the automatic mesh and camera placement behaviour of older SDKs if no Layout Manager is configured at editor time. This backwards compatible legacy mode will be removed in v10.0. As such, you are *strongly* encouraged to upgrade your scene as soon as possible.

> **NOTE**
>
> This section applies to the Unity SDK. The Native SDK Android and Native SDK iOS do not include the notion of a Display Object nor a Layout Manager (yet).

Display Object

As explained in clearvrplayer, the ClearVR SDK maintains frame-accurate synchronisation between a decoded video frame and how it is rendered to the screen. This mesh can morph on a video-frame-by-video-frame basis. Morphing can be limited to just some UVs that are changing, up to the mesh completely transforming from one shape to another (e.g. from a cube into a sphere or a quad) when switching between clips with different ContentFormats.

In pre-v9.x SDKs there was only exactly 1 (one) mesh at any point in time. This mesh would typically be automatically positioned in the "right" spot. For example, a rectilinear ("16x9") video would be positioned to appear full-screen in the app and an omnidirectional video would be positioned at the center of the camera. The SDK would take care of that for you. A downside of this mechanism was that overriding this default positioning behaviour was rather cumbersome and error prone. Since v9.0, the developer is in full control on where a Display Object will be positioned in the scene.

The ClearVR SDK provides various prefabs for the Display Objects in `Assets/ClearVR/Prefabs`.

As a convenience, you can load various placeholder meshes on a Display Object. At runtime, this placeholder mesh will be replaced by a dynamically generated mesh, but it should give you an idea on how the video will be positioned. Refer to Assets/ClearVR/Meshes for a list of available placeholder meshes.

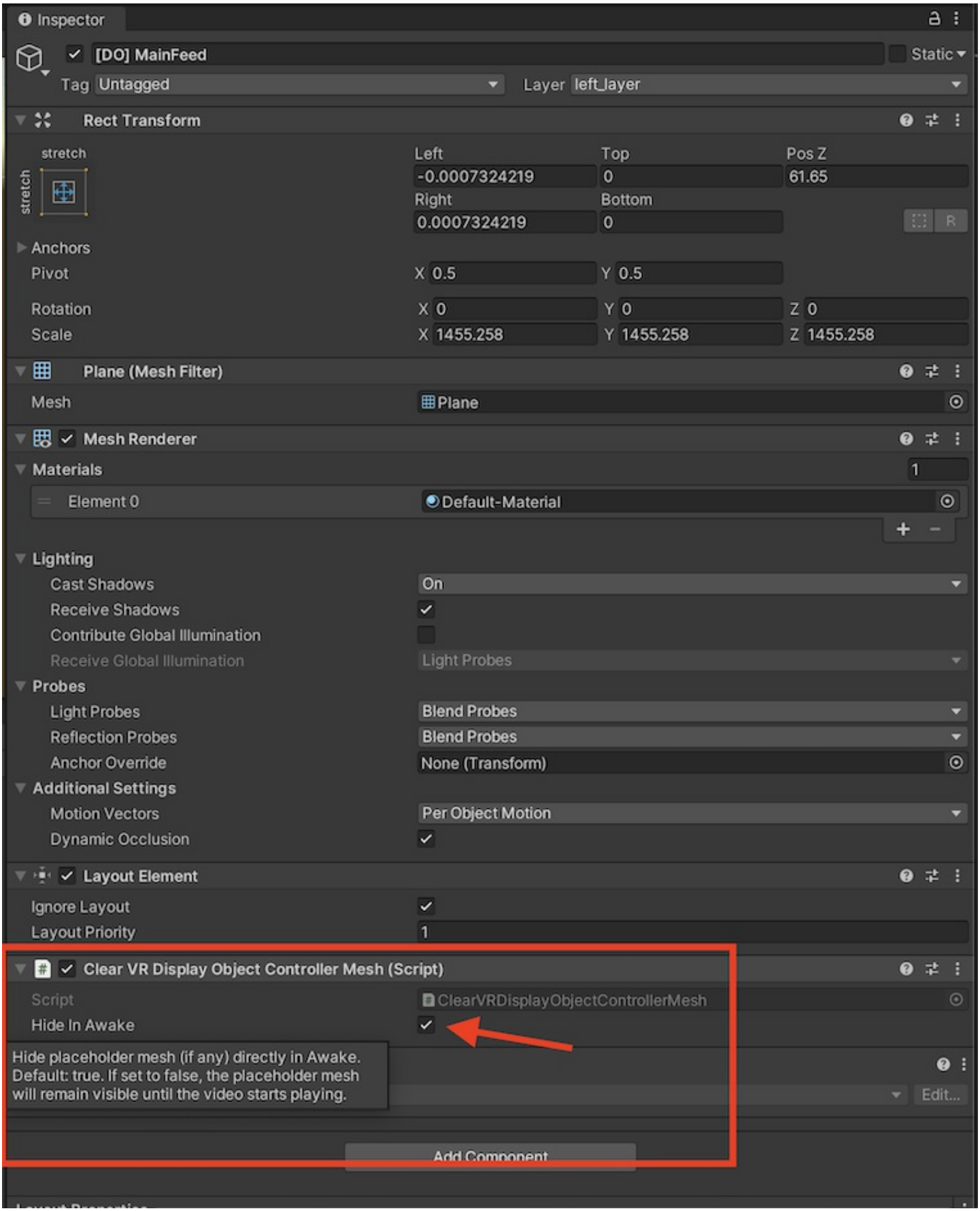
> **NOTE**
>
> The ClearVRDisplayObjectController shape, size and its behaviour are identical across all supported platforms.

Placeholder meshes

The use of a placeholder mesh is *optional*, not mandatory. The placeholder mesh can help you in designing your scene. The placeholder mesh should have similar properties as the "real" mesh that will be used when rendering the video at runtime. For example, if you would like to play a 16x9 feed on a quad, the placeholder quad mesh should be of that same aspect ratio as, on runtime, the Tiledmedia Player will generate a quad mesh of 16x9 aspect ratio automatically. If you would use a square quad placeholder mesh scaled to 16x9 via its `Transform` scale properties, at runtime that scale would be applied on top of the SDK-generated mesh (which is already of the right aspect ratio). Effectively, thsi would result in applying the aspect ratio twice yielding unexpected results.

One could use the placeholder mesh to show a loading indicator or a company logo for example. In that case, be sure to disable "Hide on Awake" tickbox on the Display Object (added in v10).

A couple of placeholder meshes are included in the SDK, see `Assets/ClearVR/Meshes`



Stereoscopic content

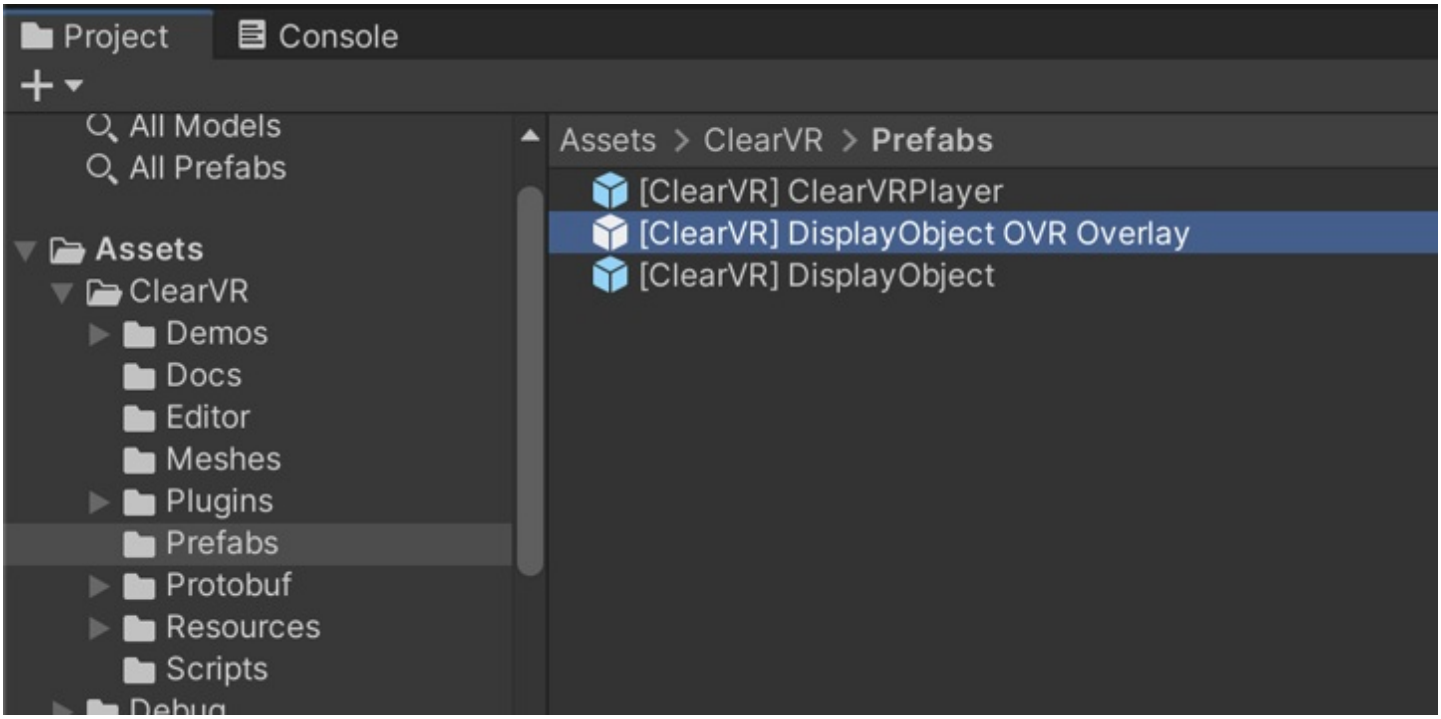
The ClearVRPlayer renders stereoscopic content on a single mesh. This mesh is rendered twice by Unity, with an UV offset between the left and right eye (Unity's mesh.uv and mesh.uv2.

Sprite support

Since v9.0, the ClearVR SDK has full support forSprite based rendering. While not so much of use when rendering omnidirectional video only, it can be of great value when designing a more 2D-oriented (e.g. canvas-driven) UI for rectilinear ("16x9") or mosaic content. Please use the provided `[ClearVR] DisplayObject Sprite` prefab as a basis.

OVROverlay support (Android only)

The `ClearVRPlayer` has full support for video playback on an OVROverlay (aka compositor layer) on Oculus Android headsets. Refer to OVROverlay for more information about what they are and what there benefits can be. The SDK includes a simple `[ClearVR] DisplayObject OVR Overlay` prefab that you can drag into your scene.



At any point in time, there can be only one ClearVRPlayer-managed OVROverlay active. Multiple OVROverlays are not supported (yet).

Due to the nature of an OVROverlay, one can not freely switch between*any* of the ContentFormats supported by the SDK. Limitations apply, please refer to the next sections for details.

NOTE

Oculus runtime version 1.59.0 or newer is required for OVROverlay-based playback. Older versions might work but are not officially supported.

NOTE

Mosaic content cannot be rendered on an OVROverlay (yet). Use non-OVROverlay based rendering for this.

WARNING

An OVROverlay rendered in Underlay mode is rendered behind the eye buffer. While convenient, the Underlay compositor layers are more bandwidth-intensive, as the compositor must punch a hole in the eye buffer with an alpha mask so that underlays are visible. Texture bandwidth is often a VR bottleneck, so use them with caution and be sure to assess their impact on your application.

ClearVR on an OVROverlay

To play ClearVR encoded content on an `OVROverlay`, you have to set `platformOptions.textureBlitMode` to TextureBlitModes.OVROverlayCopy. You can freely switch between different ClearVR encoded content (monoscopic and stereoscopic) using the `SwitchContent()` API.

> **⚠ WARNING**
>
> Widevine L1 protected playback of ClearVR content is not (yet) supported. It is only supported when playing HLS and MP4 on an OVROverlay, refer to the next section for details.

HLS and MP4 on an OVROverlay

Currently, we support the followingContentFormats for non-ClearVR (e.g. HLS and (progressive) MP4) content:

1. MonoscopicERP180
2. MonoscopicERP360
3. MonoscopicRectilinear
4. StereoscopicERP180SBS
5. StereoscopicERP360TB
6. StereoscopicRectilinearSBS
7. StereoscopicRectilinearTB

When playing other content formats (e.g. fish eye video), behaviour is undefined. This ranges from receiving an error up till your application crashing. Always check `logcat` for details.

To play HLS and (progressive) MP4 video on an `OVROverlay`, you have two options:

1. Set `platformOptions.textureBlitMode` to TextureBlitModes.OVROverlayZeroCopy. This yields the best overall performance and allows you play Widevine L1 protected content. However, you can only SwitchContent between rectilinear clips (mono/stereo).
2. Set `platformOptions.textureBlitMode` to TextureBlitModes.OVROverlayCopy. This enables you to freely switch between ContentFormats and to ClearVR content at the expense of an additional texture copy. This can have a slightly negative performance impact.

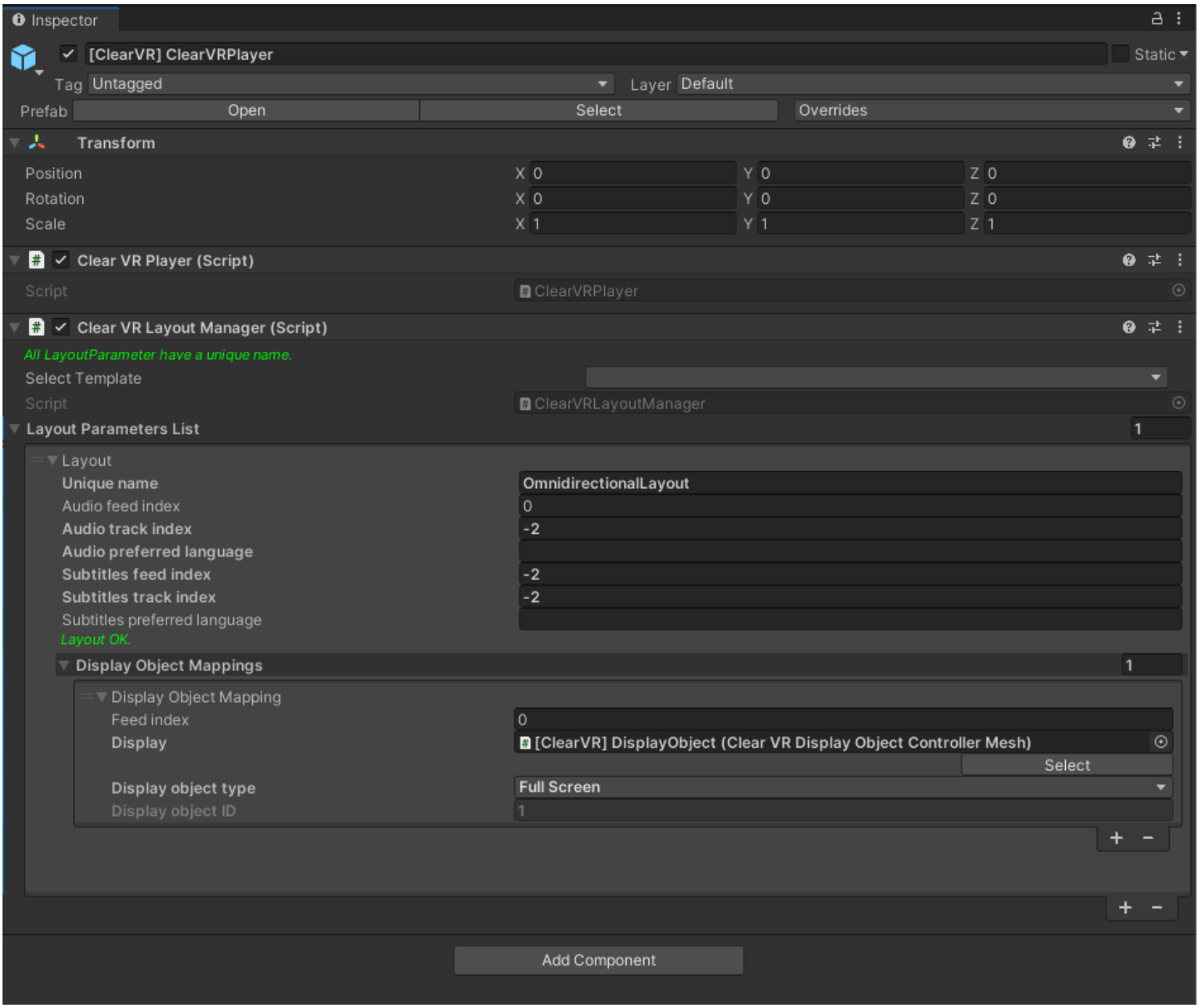
If you plan to only play rectilinear content, stick with option 1. This will apply to most developers. If you envision complex switches between rectilinear and omnidirectional content, use option 2.

Playback of Widevine L1 protected content is supported, please refer to theDigital Rights Management section for details.

> **⚠ WARNING**
>
> The SwitchContent API is available when using an OVROverlay. However, you can switch between clips with ContentFormat `*ERP*` (mono and stereo) and between clips with ContentFormat `*Rectilinear*` (mono and stereo) only. You cannot switch from e.g. `MonoscopicERP180` to `StereoScopicRectilinearSBS`. You will either receive an error might even crash. It is the resposibility of the developer to cover these corner-cases.

Layout Manager

In a single video feed environment, video positioning is typically relatively straight forward. An omnidirectional video is positioned at center of the camera (typically at position (0, 0, 0)) and classic rectilinear ("16x9") video is positioned "in front" of the user. When playing mosaic content, the setup quickly becomes more complex. It typically consists of multiple rectilinear feeds, with possibly one or multiple omnidrectional feeds in the mix as well. Each feed will end up on its own `GameObject`. The `ClearVRLayoutManager` enables you to manage your layout(s), allowing you to even seamlessly switch between the most complex constellations of feeds.



The `ClearVRLayoutManager` is typically attached to the same `GameObject` as your `ClearVRPlayer`. There shall be only one `ClearVRLayoutManager` at any point in time.

At any moment in time, you can query the current*desired* Layout Parameters of a given name inb the Layout Manager. Note that this does *not* reflect the *actual* current state of all Display Objects. If you want to know what a Display Object is showing, query its appropriate fields (like `activeFeedIndex` and `displayObjectClassType`) instead.

NOTE

If you do not add a `ClearVRLayoutManager` to your scene yourself, the `ClearVRPlayer` will add one for you automatically.

WARNING

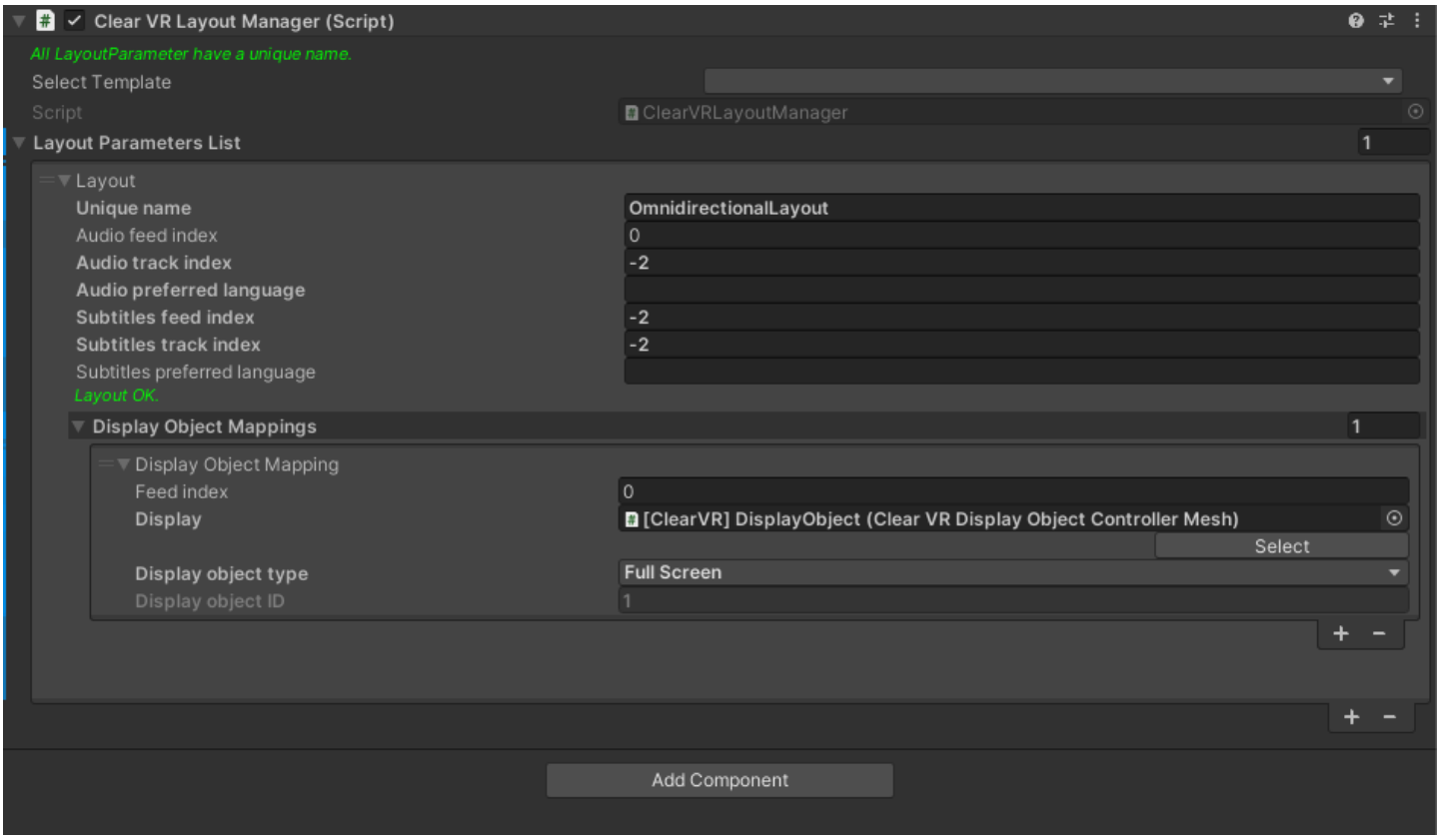
There can be only one `ClearVRLayoutManager` in your scene. Having more than one will lead to undefined behaviour, which can include DisplayObjects disappearing, showing black video or nothing at all.

Layout Parameters and Display Object Mappings

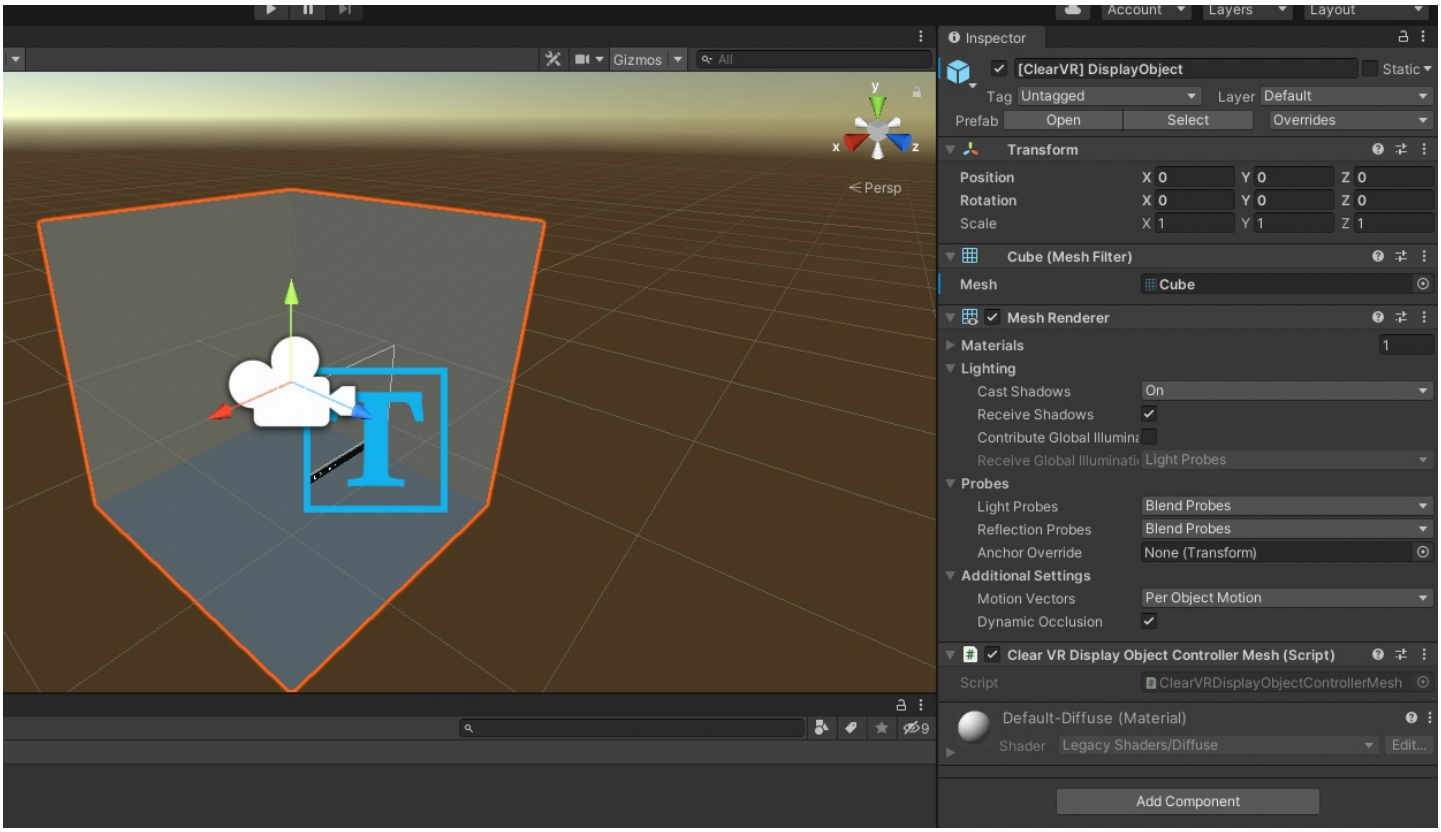
The `ClearVRLayoutManager` manages one or more `LayoutParameters`. Each `LayoutParameters` is identified by a unique name and has a list of Display Object Mappings (DOMs). A Display Object is defined as a `GameObject` that will hold one video feed. A Display Object Mapping defines the link between a Display Object, *which* feed it will show and its display object class type.

Example 1: playing omnidirectional video

When playing single feed omnidirectional (e.g. 360, 180 or fisheye) content, you will have one Display Object Mapping per `LayoutParameters`, like in below's screenshot:



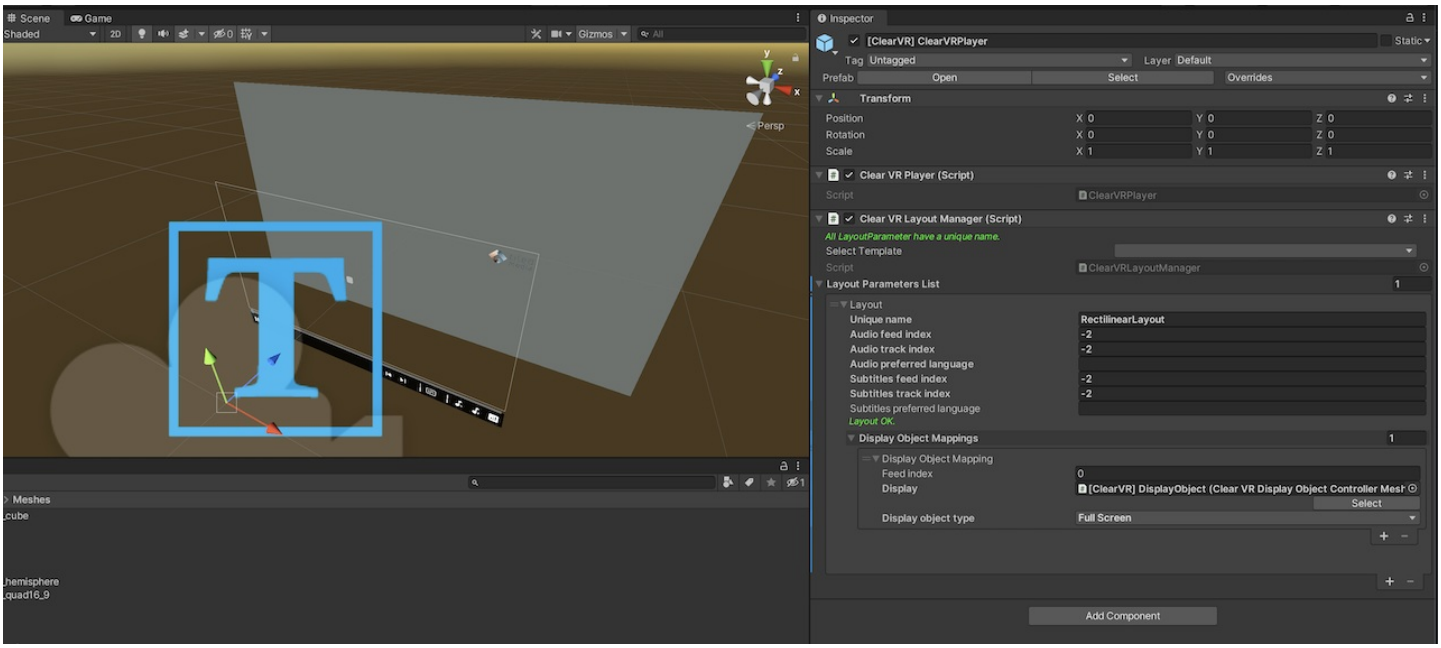
In this screenshot, we mapped feed index 0 to the `[ClearVR] DisplayObject` as a `Full Screen` feed. A cube-shaped placeholder has been used as we will play an omnidirectional 360 degree ClearVR encoded video (which uses the cubemap projection).



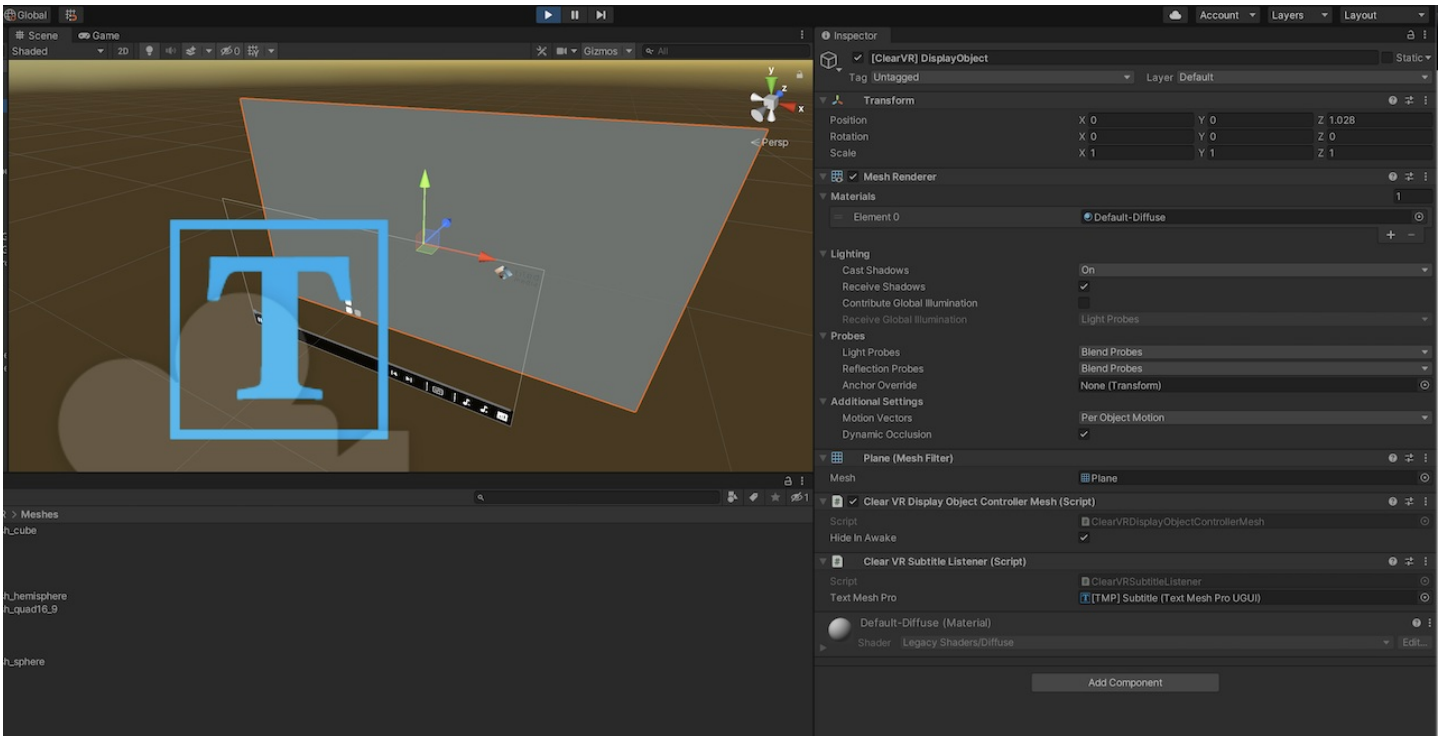
As we are configuring for omnidirectional video playback, the Display Object is located at the center of the scene (and thus in the center of the camera). Furthermore, there will be only one video feed with index 0.

Example 2: playing rectilinear ("16x9") content

When playing rectilinear (classic, "16x9") content, you will have one Display Object Mapping per `LayoutParameters`, like in below's screenshot when playing omnidirectional content:



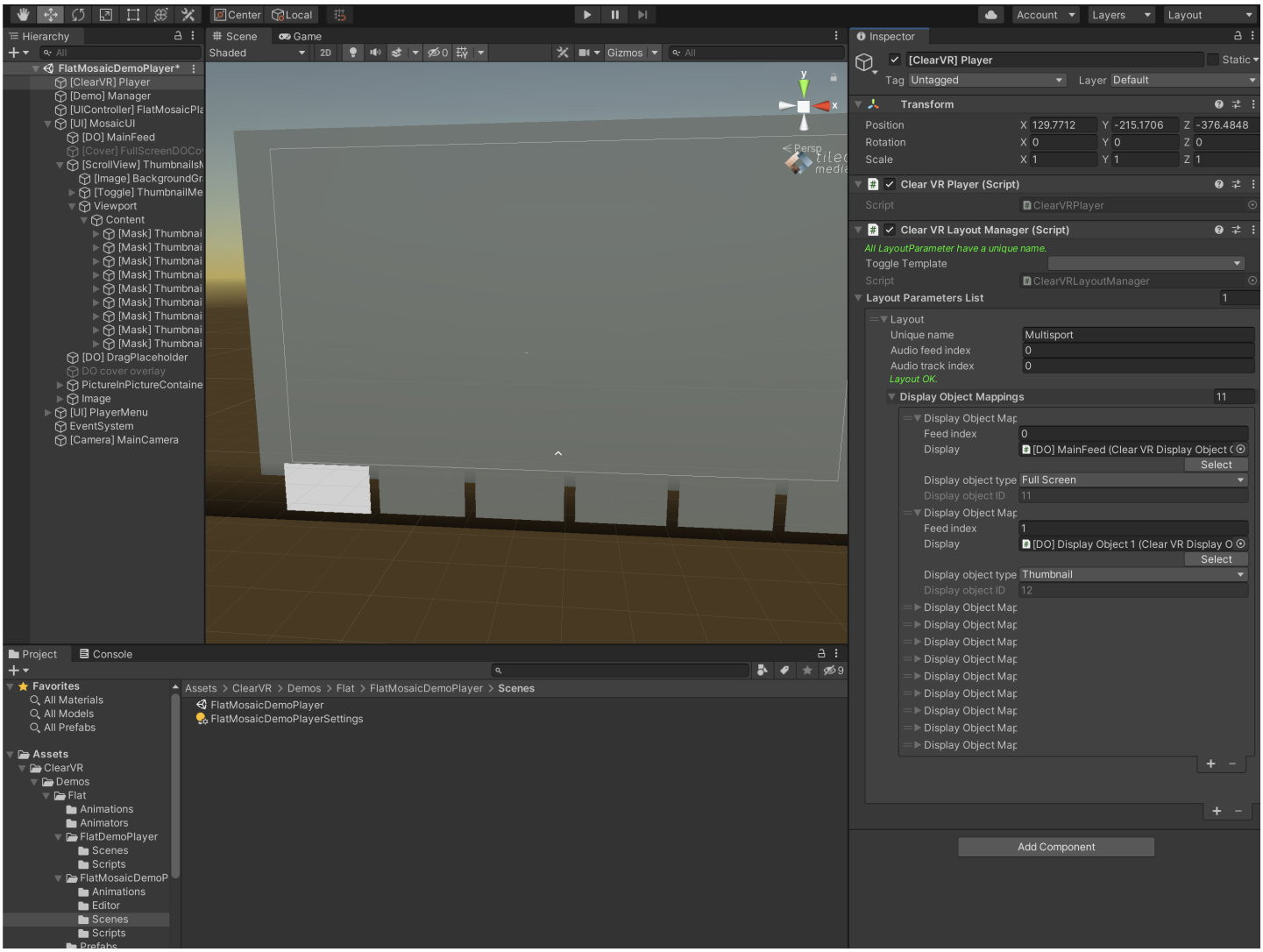
In this screenshot, we mapped feed index 0 to the `[ClearVR] DisplayObject` as a `Full Screen` feed. A cube-shaped placeholder has been used as we will play an omnidirectional 360 degree ClearVR encoded video (which uses the cubemap projection).



As we are configuring for rectilinear video playback, the Display Object is located at the appropriate z-distance in the scene. Furthermore, there will be only one video feed with index 0. Lastly, the placeholder mesh is a quad with 16x9 aspect ratio. Notice how the scale of the display object is uniform (1.0 x 1.0 x 1.0). If you expect to play content with other aspectratios (like 21x10 or 4x3), you should add appropriately constructed placeholder meshes yourself.

Example 3: playing mosaic content

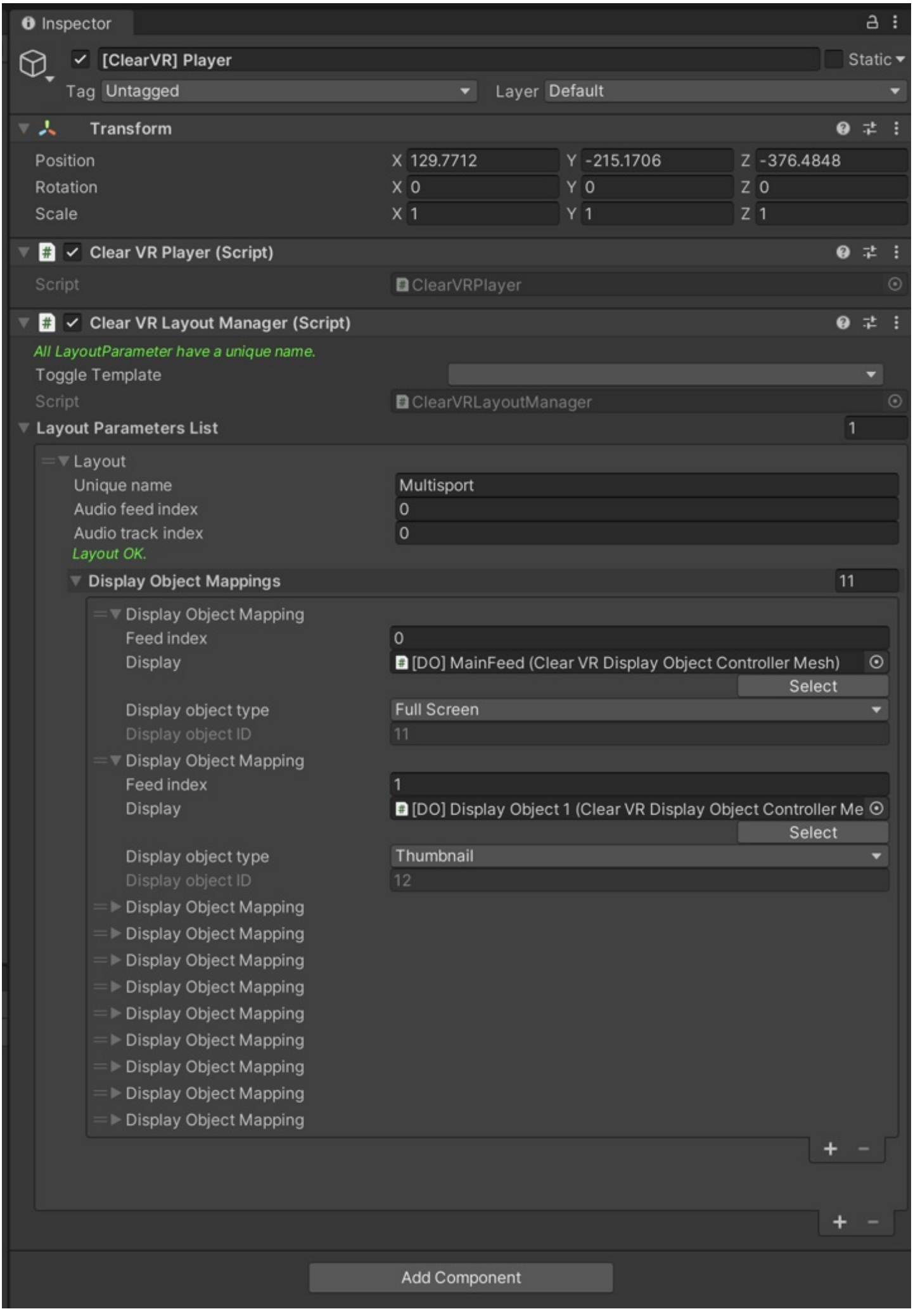
When designing your mosaic layout, it is important to have a clear idea of what video feed goes where and how to interact with them. Typically, you will have one FullScreen feed, rendering in highest available quality), 0 or more LargePanel feeds and possibly dozens of Thumbnail feeds.



Using the Layout Manager

The Layout Manager, the Display Objects and the Display Object Mappings together gives you full control over what video feed is rendered where in what form. At design time, you start by adding the `ClearVRLayoutManager` component to the same object you added the `ClearVRPlayer` component to. If you used the `[ClearVR] ClearVRPlayer` prefab, it is already added for you. Next, you position your Display Object(s) in the scene.

Back to the Layout Manager, you create a new Layout set in its Layout Parameters List. Give the new Layout a unique name, configure the audio feed index and audio track index (default values would be 0, 0 respectively to select the first audio track on the first feed) and add a Display Object Mapping to the Display Object Mappings list.


Inspector
[ClearVR] Player
Static
Tag Untagged
Layer Default
Transform
Position X 129.7712 Y -215.1706 Z -376.4848
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Clear VR Player (Script)
Script ClearVRPlayer
Clear VR Layout Manager (Script)
All LayoutParameter have a unique name.
Toggle Template
Script ClearVRLayoutManager
Layout Parameters List 1
Layout
Unique name Multisport
Audio feed index 0
Audio track index 0
Layout OK.
Display Object Mappings 11
Display Object Mapping
Feed index 0
Display [DO] MainFeed (Clear VR Display Object Controller Mesh)
Select
Display object type Full Screen
Display object ID 11
Display Object Mapping
Feed index 1
Display [DO] Display Object 1 (Clear VR Display Object Controller Me
Select
Display object type Thumbnail
Display object ID 12
Display Object Mapping
Display Object Mapping
Display Object Mapping
Display Object Mapping
Display Object Mapping
Display Object Mapping
Display Object Mapping
Display Object Mapping
Display Object Mapping
Add Component

You drag your Display Object into the `Display` field. Next, you configure the feed index this Display Object is mapped to (0 by default) and its Display Object Type (FullScreen, LargePanel, Thumbnail). The type influences the quality with which the feed is fetched from the CDN (depending on available bandwidth ofcourse). With FullScreen being the highest quality and Thumbnail being the lowest available.

If you are designing a mosaic scene with multiple Display Objects in one Layout, you can add a Display Object Mapping for each of your Display Objects.

Now that you have configured your Layout, it is time to use it in the player. In your code, you will need to tell the ClearVRPlayer which Layout should be used when initializing the player. Refer to the following code snippet:

```
public void InitializeClearVRPlayer() {
    if (clearVRPlayer != null) {
        // Check if ClearVR is supported on the current platform
        if (!ClearVRPlayer.GetIsPlatformSupported()) {
            throw new Exception("[ClearVR] Sorry, ClearVR is not yet supported on this platform!");
        }
        // Add event listener, make sure to detach the listeners when the player reached StateChangedStopped
        clearVRPlayer.clearVREvents.AddListener(CbClearVREvent);
        clearVRPlayer.clearVRDisplayObjectEvents.AddListener(CbClearVRDisplayObjectEvent);

        PlatformOptionsBase platformOptions = (PlatformOptionsBase) clearVRPlayer.GetDefaultPlatformOptions();
        platformOptions.licenseFileBytes = licenseFileBytes; /* Byte array containing your private license file data */

        // Make sure there is a main Camera as the ClearVRPlayer needs its Transform.
        if (Camera.main != null) {
            platformOptions.trackingTransform = Camera.main.transform; /* The transform of the currently active
Camera, e.g. MainCamera or the CenterEyeAnchor on your OVRCameraRig */
        } else {
            // use main camera instead if user interface is not found
            throw new Exception("[ClearVR] Main camera not found. Using first camera found in scene instead.");
        }
        // Specify ContentItem to load and preferred playout start position.
        ContentItem contentItem = new ContentItem(....);

        // Get the layoutParameters
        LayoutParameters layoutParameters =
clearVRPlayer.GetLayoutParametersByName("YOUR_LAYOUT_NAME");
        platformOptions.prepareContentParameters = new PrepareContentParameters(contentItem, null,
layoutParameters);

        // Schedule a request to Initialize a ClearVRPlayer object. Depending on the specified platformOptions, you
will receive a callback sooner or later.
        clearVRPlayer.Initialize(platformOptions,
            onSuccess: (cbClearVREvent, cbClearVRPlayer) =>
            UnityEngine.Debug.Log("[ClearVR] Player Initialized."),
            onFailure: (cbClearVREvent, cbClearVRPlayer) =>
            UnityEngine.Debug.LogWarning(String.Format("[ClearVR] Something went wrong while initializing the
ClearVRPlayer! Error Code: {0}; Message: {1} .", cbClearVREvent.message.code,
cbClearVREvent.message.message))
        );
    }
}
```

The `clearVRPlayer.GetLayoutParametersByName("YOUR_LAYOUT_NAME");` API returns a *COPY* of the layout you configured under that unique name, or null if no such named Layout was found. If you pass `null` as value for the LayoutParameters argument in the `PrepareContentParameters() constructor, the player will print a warning to the console and fallback to legacy mode where it totally ignores your Layout. As noted earlier, this legacy mode will be removed in v10.0.

Changing current Layout Parameters

NOTE

This section applies to mosaic playback only.

One of the key benefits of Tiledmedia's mosaic playback is that it allows one to instantly switch between video feeds. For example, you can instantenously switch a feed that is rendering as a `Thumbnail` with your `FullScreen` feed. You can use the `SetLayout()` API to make runtime modifications to a Layout. Various helper methods are available to facilitate this process.

```
// Swap the
public void SetFullScreenFeed(int thumbnailFeedIndex) {
    LayoutParameters layoutParameters =
clearVRPlayer.GetLayoutParametersByName("YOUR_LAYOUT_NAME"); // Returns a COPY!
    DisplayObjectMapping fullScreenDisplayObjectMapping =
layoutParameters.GetFullScreenDisplayObjectMapping(); // convenience helper to get the full screen mapping
    int index = layoutParameters.displayObjectMappings.IndexOf(fullScreenDisplayObjectMapping);
    layoutParameters.displayObjectMappings[index].feedIndex = thumbnailFeedIndex;
    clearVRPlayer.mediaPlayer.SetLayout(layoutParameters,
        onSuccess: (cbClearVREvent, cbClearVRPlayer) => Debug.Log("[ClearVR] Feeds layout changed
successfully."),
        onFailure: (cbClearVREvent, cbClearVRPlayer) => Debug.LogWarning(string.Format("[ClearVR] Something
went wrong changing feed layout! Error Code: {0}; Message: {1} .", cbClearVREvent.message.code,
cbClearVREvent.message.message))
    );
}
```

In the example above, the FullScreen feed will be swapped for the specified Thumbnail feed. The player will respond immediately by upscaling the lower quality Thumbnail feed on the FullScreen Display Object. Shortly after, depending on network conditions, the upscaled Thumbnail feed on this FullScreen Display Object will be seamlessly swapped for the high quality FullScreen feed.

NOTE

Any changes you make to the `LayoutParameters` you got through the `GetLayoutParametersByName()` API *MUST* be effectuated by calling `SetLayout()`. The API returns a *COPY*, not a reference.

Switching between Layout Parameters

If you want to switch between clips with different layouts on the fly at runtime, for example because you want to switch from a rectilinear ("16x9") clip to an omnidirectional clip, you would construct two Layouts in the LayoutManager with unique names. Subsequently, when calling the SwitchContent() API you would pass the matching LayoutParameters as an argument. The ClearVRPlayer will ensure that your videos will end-up on the correct Display Object.

Debugging issues while using the Tiledmedia SDK

When you run into unexpected behaviour, or when the observed behaviour does not match with what you would expect, the first step would be to print all ClearVREvents that you receive. If your source derived from the Tiledmedia Sample Player projects, these events are received in `CbClearVREvent(...)`. Add a print statement at the beginning of this callback to get a better understanding of what is happening. For convenience, you can use the following line:

```
private void CbClearVREvent(MediaPlayerBase mediaPlayer, ClearVREvent clearVREvent) {
    clearVREvent.PrintShort()
    ...
}
```

Gathering verbose Tiledmedia SDK logs

Sometimes, we ask you to temporarily enable verbose Tiledmedia SDK logging when you run into an unforeseen problem. To enable verbose logging, add the following lines to your code *before* you set-up your PlatformOptions:

```
LoggingConfiguration loggingConfig = LoggingConfiguration.GetDefaultLoggingConfiguration()
ClearVRPlayer.EnableLogging(loggingConfig);
```

Notice that this is a `static` API, which can be called at any point in time.

This will write verbose logging (`.tmlog`) and event recorder files (`.tmerp`) to disk. The exact output folder is written to console/logcat after the `ClearVRPlayer.EnableLogging()` API call. In Unity, the default folder equals `Application.persistentDataPath` on all platforms.

> **⚠ WARNING**
>
> Enabling verbose logging incurs a (significant) performance penalty. Be sure to *never* enable this logging by default in your production application.

Uploading Tiledmedia SDK logs for analysis by Tiledmedia

To facilitate customer-side debugging of potential application and/or Tiledmedia SDK issues, besides enabling verbose logging, v10.0 adds the option to upload gathered log files to the Tiledmedia backend. To use this feature, one would:

1. Enable verbose logging (see the previous section).

2. Trigger the problem.
3. Call the UploadLogs API, you can do this e.g.:
    - as soon as the problem has happened (e.g. in a callback or after a specific `ClearVREvent`).
    - at the end of the run (e.g. after you have reached `StateChangedStopped`).
    - at the beginning of the next run (for example if the app or the Unity Editor crashed because of the problem you ran into).
4. Share the unique upload id with Tiledmedia.

Example code:

```
ClearVRPlayer.UploadLogs(YOUR_LICENSE_FILE_BYTE,
    onSuccess: (uniqueUploadID, optionalArguments) => {
        UnityEngine.Debug.Log("Tiledmedia log upload success. Unique upload id: " + uniqueUploadID);
    },
    onFailure: (clearVRMessage, optionalArguments) => {
        UnityEngine.Debug.Log("Tiledmedia log upload failed. Error details: " + clearVRMessage);
    }
);
```

If you do not want to use this API, you can also gather the log files yourself and share them as a ZIP package with Tiledmedia.

Acknowledgements

A special thank you goes out to the people that have helped making our technology better:

- Kevin
- Emmanuel
- Kobi
- Andy
- Nick
- Ben
- Mikal
- Blake
- Denis
- Vincent
- Olie
- Xavier
- Jani
- Joey
- Dmitry
- Ivan
- Pushpinder
- KD
- Dailos
- Osmany Rodríguez
- Yury
- Hyunwook
- Ellie

Changelog

Overview of changes per release.

v10.1.3 (2023-08-22)

Cross-platform

- Fix issue where a specific subtitle payload could result in a crash.

Unity SDK

- Fix issue where request callbacks were not triggered. This regression affected v10.1.1 and v10.1.2 (#7087).

v10.1.2 (2023-08-21)

Cross-platform

- Improved synchronization of subtitles (#7014).
- Fix issue where a `${DUMMY}` subtitle sometimes showed up (#7074).
- Fix issue where the player could get stuck in infinite buffering if a fail-over occurred.

v10.1.1 (2023-08-17)

Cross-platform

- Fix rare crash in `MeshAndTextureManager::DestroyTexture()`, which was sometimes triggered during an application suspend/resume cycle (#6889)
- Fix issue where content with subtitles would not stop playint on very long-running live streams (#7061).

Unity SDK

- Fix issue where the player would not complete a `Stop()` request if it was requested*just* after `Initialize()` was called (#6656).
- Fix rare `NullReferenceException` in `GetTransformationMatrixFromMeshDescriptionStruct()` (#6997).
- PC: fix issue on Nvidia and AMD discrete graphics cards where a green flash frame could appear when switching between clips encoded with b-frames (#6851).

Native SDK Android

- Fix `NullReferenceException` in `TMLogger` component when the library would attempt to log something when called from a `View` constructor before an Activity's `onCreate()` had completed.

Native SDK iOS

- Fix issue where player could get stuck when hammering application suspend/resume at super-human speed (#6905, #7011).

v10.1.0 (2023-08-07)

> **⚠ WARNING**
>
> For customers targeting the iOS/ipadOS/tvOS platforms (native and Unity): this SDK includes a mandatory fix for iOS 17 support. ClearVR and mosaic video will show up black in your apps running on OS version 17 on SDK versions prior to v10.1.0.

Unity SDK

- iOS: fix issue in bitstream rewriting where value of `no_output_of_prior_pics_flag` was incorrectly set for 'grey tiles', leading to playback issues on iOS 17 beta (#6935).
- iOS: fix issue where some video frames would sometimes not be displayed when changing feed layout. Visually, this would result in a short hick-up in the video (#6872).
- Fix issue where `ResumingPlaybackAfterApplicationRegainedFocus` was not triggered after an application Suspend/Resume cycle (#6869).
- Fix issue where a `DisplayObject` would stop showing video if it was part of a `LayoutParameters` set, removed from that set and later re-added again (#6926).
- Fix issue where destroying a `DisplayObject` in any `ClearVRDisplayObjectEvent` event (e.g. `FirstFrameRendered`) would result in an `IndexOutRange` exception (#6955).
- Android: add support for auto-detecting `Wave VR` and `Skyworth VR` XRPlugins (#6948).
- Android: re-enable stereoscopic video rendering on `Wave VR`-based headsets (#2245).

Native SDK iOS

- Fix issue where some video frames would sometimes not be displayed when changing feed layout. Visually, this would result in a short hick-up in the video (#6872).
- Fix issue where a `TMClearVRUIView` could end-up in an invalid state if it was created*and* destroyed before a ClearVRPlayer was initialized.
- Fix issue in bitstream rewriting where value of `no_output_of_prior_pics_flag` was incorrectly set for 'grey tiles', leading to playback issues on iOS 17 beta (#6935).

v10.0.6 (2023-06-20)

Cross-platform

- Fix issue where a feed in a mosaic stream would not always switch to a higher ABR rendition (affected VR headsets only).
- Fix issue where the last content position was not properly restored after an application suspend/resume cycle when a clip reached the end (with looping disabled) (#6762).
- Fix issue where `initialAudioGain` was not always maintained when quickly suspending and

resuming your application. This only affected apps setting a non-default `initialAudioGain` (#6763).
- Fix issue where the mute state was not properly maintained between suspend/resume cycles (#6773).
- Fix issue where the `GetTimingReport(WallclockTime)` did not always return position and duration in wallclock time reference frame (#6798).

Unity SDK

- Android: fix possible compounding performance slow down after application suspend/resume cycles.
- Android: fix issue where recent versions of the PicoXR Unity XR Plugin were not detected due to an undocumented API change in their SDK. This resulted in incorrect viewport tracking for ClearVR content (#6814).
- Android: force video decoder cold reinit when framerate changes on specific device to prevent audio and video stutter (#6373)
- PC: fix jumping frames while seeking during paused with LibAV decoder (#6635).
- PC: fix potential hangs when closing the editor after initialize failed (#6651).
- Sample player: fix issue where another UpdateUI loop was started after each application suspend/resume cycle without stopping the previous one (#6764).
- Fix issue where Tiledmedia Player SDK logging, if explicitly enabled for debugging purposes, was sometimes incorrectly configured resulting in some of its components to log with an incorrect log level (#6606).
- Automatically fallback to a platform specific generic HMD device type if an XR Plugin powered headset is running but not recogonized by the Tiledmedia Player SDK (#6825).
- Windows: fix issue where Adaptive Bitrate Bandwidth Estimator was not working on Windows.

Native SDK Android

- Fix possible compounding performance slow down after application suspend/resume cycles.
- Force video decoder cold reinit when framerate changes on specific device to prevent audio and video stutter (#6373).

v10.0.5 (2023-05-19)

Cross-platform

- Fix issue where some SDK log lines could get lost during player initialization and destruction (#6679).
- Fix issue where when enabling a new logging configuration from Unity, the new setup would not be considered in MF logs (#6606).

Unity SDK

- Calling `DisplayObjectController::ToString()` before it was initialized resulted in a `NullReferenceException` (#6703).

- Android: fix issue where playback could fail on Samsung S8 and Samsung S9 devices due to incorrectly reported MediaCodec capabilities (#6680).
- Android: add support for GSXR XR-Plugin-powered headsets (#6687).
- Android: fix rare `NullReferenceException` on Google Pixel 3(a) devices when seeking in video clips with b-frames (#6473).

Native SDK Android

- Fix issue where playback could fail on Samsung S8 and Samsung S9 devices due to incorrectly reported MediaCodec capabilities (#6680).
- Fix rare `NullReferenceException` exception on Google Pixel 3(a) devices when seeking in video clips with b-frames (#6473).

v10.0.4 (2023-05-10)

Cross-platform

- Fix rare issue where playback could stop when seeking outside the segment availability window in live streams if the individual feeds in the mosaic originated from different encoders (#6665).

v10.0.3 (2023-05-05)

Cross-platform

- Fix rare problem related to switching content between a non-DRM protected clip and a DRM protected clip where the audio and video track are both protected.
- Fix issue where `VideoTrackInfo` did not contain the correct aspect ratio if a feed contained padding.
- Fix very rare crash when content loading got cancelled before it completed (#6631).

Unity SDK

- Fix
  `ClearVRPlayer.cs(1447,32): error CS0136: A local or parameter named 'optionalArguments' cannot be declared in this scope`
  error when targetting .NET Standard 2.0 (#6641).

v10.0.2 (2023-04-26)

Unity SDK

- Fix issue where video decoder capabilities were underreported, causing some mosaic clips and high resolution omnidirectional content to no longer play in the Editor.

v10.0.1 (2023-04-24)

Cross-platform

- Fix rare crash when playing a DRM clip with multiple DRM protected tracks.

v10.0 (2023-04-21)

Cross-platform

- Improved ultra-low-latency support.
- Add subtitle support for HLS content with webvtt subtitles (#5817)
- Upgrade logger for Android and iOS libraries.
- Fixed issue when applying any resampling on audio decoded samples in iOS (#6280).
- Speed-up content loading by reducing the number of API calls and HTTP calls involved (#5061).
- Expose eventType (Live, FinishedLive or VOD) on TimingReport (#6442).
- Expose `name` on `SubtitleTrackInfo` in `ContentInfo` (#6537).
- Make `libspatialaudio.so` loaded at runtime in the Android platform.
- add support for telemetry reporting.

Unity SDK

- Add `ContentInfo::GetSelectableAudioTracks()` and `FeedInfo::GetSelectableAudioTracks()` convenience APIs (#6187).
- The `displayObjectMapping.displayObjectID` property is no longer publicly available. The ID cannot be relied upon to be constant between runs and, as such, cannot and should not be used as an *unique* identifier (#6561).
- Print an error message when more than one `ClearVRLayoutManager` is found in the scene. You can only have exactly one per scene. Having multiple will cause undefined behaviour, Displayobjects not showing any video or the wrong video (#6214).
- While loading content, display objects with a placeholder mesh now automatically disappear from the very first frame by default, and will show up once the feed for that display object is loaded. Previously, the placeholder mesh would be visible for a brief instant in time. This default behaviour can be overridden per display object.
- Editor: fix issue where the wrong player APIs were invoked when running in the Unity Editor while BuildTraget was set to iOS or tvOS (#6390).
- iOS: fix issue where `MediaFlow.framework` was added over and over again to the linking phase if "Append" builds were created (#6512).
- iOS: allow playback rate controlled playback synchronization (#6533).

Native SDK Android

- Add cardboard support

Native SDK iOS

- Add multiview support
- Add cardboard support
- Allow playback rate controlled playback synchronization (#6533).

v9.1.3-1-g262f2add (2023-04-11)

Cross-platform

- Fixed issue where switching from a non-DRM to a DRM clip while the audio samplerate stayed the same would result in corrupted audio.

Unity SDK

- Fix issue where a `LayoutParameter`'s `audio feed index` and `audio track index` would default to 0 instead of -2 if added via the Inspector GUI. The latter behaviour means: "select the most logical audio track automatically", while the former means: "unconditionally play the audio from feed 0, track 0". This would fail in case a clip does not have any audio (on feed 0) (#6173).
- Renamed `VideoTrackInfo[] FeedInfo::GetActiveVideoTracks()` API to `VideoTrackInfo FeedInfo::GetActiveVideoTrack()` as there can be at most only one active `VideoTrack` per `Feed`.
- Add helper API to `ContentInfo` and `FeedInfo` structs to easily access the `VideoTrackInfo` for a given `DisplayObject` (#6158).
- Android: only query DRM subsystem support when its UUID != 0x0. This can speed up player initialization by up to 4 seconds in the rare cases where `MediaDrm.isCryptoSchemeSupported()` is not behaving according to the Android specifications (#6207).
- Fix issue where the `TrackID` constructor was not publicly available (#6267).
- Fix issue where the `UnityEngine.XR` namespace was not imported in `Utils.cs` on Unity 2019.1+, but only in Unity 2020.1+. The SDK uses the namespace already on 2019.1 (#6307).
- Fix issue where a `NullReferenceException` could be thrown in some versions of Unity 2019.4 in `ClearVRLayoutManagerEditor::CheckIfLayoutParametersListSizeChanged` (#6339).
- Fix issue where `VRMosaicDemoPlayer` Unity scene was missing its `LayoutParameter` configuration in the inspector (#6376).
- iOS: the `(xc)framework` is no longer built with bitcode enabled, as Apple hasdeprecated support for it in XCode 14.

Native SDK iOS

- Fix issue where the `ClearVRPlayer` object could sometimes leak if not dereferenced in a specific way.
- The `xcramework` is no longer built with bitcode enabled, as Apple hasdeprecated support for it in XCode 14.

v9.1.2-1-g40248233 (2022-12-06)

Cross-platform

- The default audio gain has been raised from 0.5 to 1.0 (on a scale of [0.0, 1.0]) (#5987).
- Work-around issue where generated HEVC bitstream was not always in line with decoder level specifications (#5941).
- Fix issue where `controller.SetAudioGain()` was ignored if no audio was playing (yet). You can now call this API after `ClearVREventTypes.ContentPreparedForPlayout` (#6051).

Unity SDK

- The LayoutManager's `Toggle Template` drop down menu no longer activates/deactivates the DisplayObjects, but rather selects/deselects them in the hierarchy (#5932).
- Add `VideoTrackInfo::aspectRatio()` API (#5958).
- Android: fix texture memory leak on some devices when one explicitly set `platformOptions.textureBlitMode = TextureBlitModes.UVShufflingCopy`, which is mandatory on Android when using Sprite meshes (#5945).
- Android and iOS: fix issue where StateChangedStopped could be received twice if the player experienced a particular error (#6003).
- Android and iOS: fix issue where calling `clearVREvent.message.ParseAudioTrackChanged()` could return unexpectedly `null` if the application was running on a device that had its LOCALE configured to treat a `,` as a thousands separator instead of a decimal separator (#6013).
- iOS: fix memory leak in audio decoder (#6056).
- PC: fix issue where setting audio gain or mute/unmute do not have any effect (#5996).
- PC: fix potential crashes in DX11 mode after a playback stop/start cycle (#6008).
- PC: fix potential crashes on Windows without an Nvidia graphics card (#6041).
- Documentation: fixed various bugs in ClearVRPlayer.applicationUnpausedDelegate and .applicationRegainedFocusDelegate sample code.

Native SDK Android

- Add `VideoTrackInfo::getAspectRatio()` API (#5958).

Native SDK iOS

- Add `TMVideoTrackInfo::aspectRatio` API (#5958).
- Fix memory leak in audio decoder (#6056).

v9.1.1-1-g2d2f5615 (2022-11-03)

Cross-platform

- Fix issue where calling `mediaInfo.GetContentInfo()` could sometimes unexpectedly return `null` if `null` was passed as argument (#5853).

Unity SDK

- Fix Unity 2018 compatibility issues. Please note that the *Unity Sample Player project* requires

Unity 2020.3.21+ (#5865).

- Fix TracKID.Equals() not always returning the correct value (#5904).
- Fix rare `InvalidOperationException: The LinkedList is empty.` crash in `MediaPlayerBase.UpdateClearVREventsToInvokeQueue()` (#3550).
- Android: fix crash when starting playback on Unity 2018.4 (#5933).
- Android: fix `No static method metafactory` crash in `VideoFrameVSyncHelper.start()` when targeting Java `VERSION_1.7` or older (#5939).
- iOS: reduce peak memory pressure.
- iOS: fix rare random crash when playing b-frame content that does not start with an I-frame (#5857).
- iOS: fix rare crash when quickly pushing an app to the background and foregrouding again and again (#5879).

Native SDK iOS

- Reduce peak memory pressure.
- Fix issue where fish eye content was not properly configured when using `TMFishEyePresets` (#5872)
- Fix rare random crash when playing b-frame content that does not start with an I-frame (#5857).
- Fix rare crash when quickly pushing an app to the background and foregrouding again and again (#5879).
- Fix rare crash upon getting a 404 when trying to play content that wasn't there. (#6003)

v9.1-1-g322abae7 (2022-10-12)

Cross-platform

- Breaking: add `EventTypes::FinishedLive` event type to distinguish VOD from Finished Live clips. Be sure to update your logic that depends on the `EventType`.
- Breaking: the `EventTypes::SuspendingPlaybackBeforeApplicationPaused` has been renamed to `SuspendingPlaybackAfterApplicationLostFocus` and `EventTypes::ResumingPlaybackAfterApplicationPaused` has been renamed to `ResumingPlaybackAfterApplicationRegainedFocus` to make it clearer when they are fired exactly.

Unity SDK

- Fix issue where in rare cases the `onFailure` callback was not triggered on `clearVRPlayer.Initialize()` (#5747-A).
- Android: fix issue where player initialization would silently fail if none of the `LayoutParameters` in the `LayoutManager` had any Sprite Display Object configured and Sprite Display Objects were added on runtime to the `preapreContentParameters.layoutParameters` (#5747-B).
- Added `eventType` field to `ContentInfo` and `supportedStatus` field to `VideoTrackInfo` (#5802, #5803).
- The `mediaInfo.GetEventType()` has been deprecated and replaced by `mediaInfo.GetContentInfo()`.

- Add `clearVRPlayer.applicationUnpausedDelegate` callback, to override player's default playback-resume behaviour when the application is unpaused (#5838).
- Fix issue where `mediaInfo.GetContentInfo(ContentItem)` could not be called with a null argument, despite the documentation claiming that you can do that (#5837).
- Fix rare null reference exception when switching from Sprite based rendering to non-sprite based rendering (#5768).
- Allow setting `layoutParameters.audioTrackID = null`. which is interpreted as: use default values. This is in line with the `LayoutParameters` constructor (#5764).
- Fix null reference exception when print a `DisplayObjectMapping` that references a `DisplayObjectController` that is attached to a `gameobject` that has been destroyed (#5774).
- Fix issue where a dynamically spawned `DisplayObject` could sometimes reuse a still-in-use `DisplayObjectID`, resulting in the new `DisplayObject` to not show any video (#5776, #5777).
- Renamed `clearVRLoadPreviousStateDelegate` to `applicationRegainedFocusDelegate` callback. The new callback now includes a third `ContentInfo` argument.
- Android: fix issue where one frame was sometimes rendered out of order after seeking in a clip with b-frames (#5796, #5607, #5767).
- Android: fix issue where the deviceType was always detected as `AndroidPicoVRGeneric` in hybrid apps that target both flat and PicoVR when using the PicoXR SDK (#5648).
- Android: fix issue where `platformOptions` verification would fail if there was a Sprite mesh configured in any of your `LayoutParameters` and `textureBlitMode` was *not* set `Default` or `UVShufflingCopy`, even if that Sprite mesh was not used at all on runtime (#5791).
- Android - OVROverlay: fix issue where rectilinear video was not always rendered on the screen and add support for rectilinear video in TextureBlitModes.OVROverlayCopy (#5758).
- Android - OVROverlay: fix issue where ultra wide (21:9) stereoscopic top-bottom rectilinear video was interpreted as being side-by-side, causing skewed playback (#5758).

Native SDK Android

- Added `eventType` field to `ContentInfo` and `supportedStatus` field to `VideoTrackInfo` (#5802, #5803).
- Fix issue where one frame was sometimes rendered out of order after seeking in a clip with b-frames (#5796, #5607, #5767).

Native SDK iOS

- Breaking: `TMContentTypes.Vod` and `.Live` have been renamed to `.vod` and `.live` respectively.
- Added `eventType` field to `ContentInfo` and `supportedStatus` field to `VideoTrackInfo` (#5802, #5803).

v9.0-1-ge82714aa (2022-09-14)

This is a major release with a number of breaking API changes, especially related to (automatic) mesh placement. Also, it adds full support for plyback of multi-stream (mosaic) content. Customers upgrading their project to this new release, you are strongly encouraged to check the upgrading to v9 section for important details.

Cross-platform

- The following APIs, which take a single callback to handle both success and failure, have been deprecated in favor of similarly named APIs that have a two callbacks (`onSuccess` and `onFailure`):
    - `StartPlayout()`, `Pause()`, `Unpause()`, `TogglePause()`, `Stop()`, `Seek()`, `SwitchContent()` & `SetAudioTrack()` on the `controller` interface of the `ClearVRPlayer` class.
    - `PopulateMediaInfo()`, `PrepareContentForPlayout()` and `SetStereoMode()` on the `mediaPlayer` interface of the `ClearVRPlayer` class.
    - `EnableSync()`, `DisableSync()` and `PollSyncStatus()` on the `sync` interface of the `ClearVRPlayer` class.
    - `Initialize()` and `TestIsContentSupported()` on the `ClearVRPlayer` class.
- `FishEyeCameraAndLensTypes` enum is renamed to `FishEyePresets`.
- Add support for playback of non-progressive MP4 clips.
- Many performance improvements.

Unity SDK

- Breaking change: you now have full control over the position of the GameObject that will render your video(s). Fallback to automatic GameObject placement is provided, but will be removed in a next release.
- Breaking: `contentItem.startViewportAndObjectPose` has been removed and can no longer be used. Refer to the new (Layout Manager)[layoutmanager.md] documentation for details on how to upgrade.
- Breaking change: if a callback is specified in a request with associated `ClearVREventType`, both the callback *and* the associated event are fired. Previously, only the callback was fired. For example, calling `controller.Stop(callback, callback)` will trigger both the callback and `StateChangedStopped` now (#5220).
- Breaking change: iOS: drop support for OpenGLES on all Unity versions.
- Android: add support for low latency video decoders which can be available on devices running API 30 and up (#4104).
- Android: full support for `OVROverlay` based video rendering of ClearVR, ERP and rectilinear ("16x9") content on Oculus headsets.
- Improved support for XR-plugin-based third party libraries for PicoVR, SkyworthVR and other headsets.
- Support for `Sprite`-based rendering of non-omnidirectional video.
- PC: the RTAudio dependency was bumped to v5.2.0 (#5139).
- PC: fix issue where Sample Player couldn't handle the Nvidia reaching-out-of-bound segfault then crash (#3961).

Native SDK Android

- Add support for low latency video decoders which can be available on devices running API 30 and up (#4104).
- Various performance improvements and bug fixes.

Native SDK iOS

- Add support for the latest iOS devices.
- Add support for simulator and AppleTV.
- Various performance improvements and bug fixes.

v8.2.5-1-g954fe51f (2022-06-07)

Unity SDK

- Android: fix issue where a tethered PicoVR headset was always automatically detected, even if it was not connected (#5496).

v8.2.4-2-g548ec84b (2022-06-24)

Cross-platform

- Fixed issue where player would get stuck when seeking in an HLS stream to a ContentTime beyond the last IDR frame but before the end of the clip.

v8.2.3-1-gec14f6f5 (2022-05-04)

Cross-platform

- Fix issue where file:// playback was broken

Unity SDK

- iOS: fix issue where the player state was not correctly restored after an app suspend/resume cycle (#5343).

v8.2.2-1-g372b1ac3 (2022-04-15)

Cross-platform

- Fix issue where viewport and display object position/orientation are not restored during suspend/resume cycle (#5298).

v8.2.1-1-gfcbec6c9 (2022-04-11)

Cross-platform

- Fix issue where consecutive Seek requests could result in slow adaptive bitrate switching (#5108).

- Fix issue where ABR content using meta-manifests and multiple tracks per bundle would cause content to never start playing (#5281).
- Fix issue where certain HLS feeds would throw `manifest doesn't contain any segments` error while there were actually segments in that manifest.
- Fix issue where TCP issues during network requests could lead to buffering unnecessary repeated buffering (#5281).

Unity SDK

- Renamed: `TransitionTypes.Continous` has been renamed to `TransitionTypes.Continuous` (#5187).
- Fix `The variable 'isPVRLibraryfound' is assigned but its value is never used` warning in Unity Editor (#5110).
- Fix issue where `SwitchContentParameters.transitionType` argument always defaulted to `Fast` (#5186).
- Fix issue where `Main Color` component was not used in the `ClearVROmniShader` fragment shader (#5244).
- Fix issue where a custom `ContentItem.startViewportAndObjectPose` was not preserved during an app suspend/resume cycle (#5278).
- Fix issue where playback would not resume if app was suspended just after `clearVRPlayer.Initialize()` was called (#5289).
- Android: when enabled, SDK logs will now be stored in `Application.persistentDataPath` on Android API 29+ to handle Scoped Storage file access limitations (#5076).
- Android: fix `You cannot call getIsSecureDecoderRequired() before calling initializeMediaDrm().` crash when switching to a `ContentItem` that has a valid `DRMInfo` struct but which was not DRM encrypted (#5275).
- PC: fix issue where audio playback could break if playback rate was set to != 1.0 (#5074).
- iOS: fix issue where the callback of ClearVRPlayer.SwitchContent() would not be triggered in case of a successful switch (#5123).

Native SDK Android

- Renamed: `TransitionTypes.Continous` has been renamed to `TransitionTypes.Continuous` (#5187).
- Fix `You cannot call getIsSecureDecoderRequired() before calling initializeMediaDrm().` crash when switching to a `ContentItem` that has a valid `DRMInfo` struct but which was not DRM encrypted (#5275).
- Fix issue where playback would not resume if app was suspended just after `clearVRPlayer.Initialize()` was called (#5289).

Native SDK iOS

- Fix issue where the callback of ClearVRPlayer.SwitchContent() would not be triggered in case of a successful switch (#5123).
- Fix issue where playback would not resume if app was suspended just after `clearVRPlayer.Initialize()` was called (#5289).

v8.2-7-g77a31c58 (2022-02-10)

Cross-platform

- Add support for Widevine DRM-encrypted mosaic streams
- Add support for Widevine DRM-encrypted audio tracks
- Add support for out-of-order B frames in mosaic streams, improving quality-per-bit
- Reduce memory usage when playing content with extremely long segment sizes (i.e. segments of a minute each)

Unity SDK

- Add new `ClearVREventTypes.AudioFocusChanged` event. The player will now receive callbacks when audio focus has been lost or regained (#4996).
- Fix issue where the preferred audio track language code was not maintained after an application suspend/resume cycle (#5017).
- Fix issue where an incorrect audio track was sometimes selected when playing a multi-feed `ContentItem` (#4823).
- Fix issue where if a `clearVRLoadPreviousStateDelegate` was registered, a NullReferenceException would occur when an application suspend/resume cycle was triggered while initializing the ClearVRPlayer (#4997).
- Editor and Standalone: fix issue where `GetTimingReport(TimingTypes.ContentTime)` returned an all zeros `TimingReport` (#5002).
- Android: add support for OVROverlay-based video playback of non-ClearVR encoded content (e.g. HLS or MP4) on Oculus Android headsets (#1062).
- Android: add support for playback of Widevine L1 encrypted content using OVROverlays on Oculus Android headsets (#1062).
- Android: fixed typo: `OVROVerlayOptions` was renamed to `OVROverlayOptions`.
- Android: enable support for playback of Widevine-encrypted AVC content (#4954).
- Android: add support for Widevine encrypted audio tracks (#4955).
- Android: fix issue where in rare cases shuffled tiles could be visible for a brief instant during heavy system load (#4976).
- Android: add support for low latency video decoders, which can be available on devices running API 30 and up (#4104).
- iOS: fix issue where frame judder could be observed when playing content with a deep b-frames GOP structure (#5014).
- iOS: fix issue where audio would be silenced for the rest of the session after being called. (#4996)
- iOS: fix issue where the application would segfault in the rare cases that two subsequent video frames have identical RTS (#5003).

Native SDK Android

- Upgrade sample project to Android Studio 2020.3.1 Arctic Fox. Bumps `targetSdkVersion` from 28

to 31 and gradle to v7.0.4.
- Enable support for playback of Widevine-encrypted AVC content (#4954).
- Add support for Widevine encrypted audio tracks (#4955).
- Fix "Got an unexpected value when switching stereo mode..." warning (#5004).
- Fix issue where in rare cases shuffled tiles could be visible for a brief instant during heavy system load (#4976).
- Fix issue where `StateChangedStopped` was sometimes never triggered if the `ClearVRTextureView` was destroyed before `controller.stop()` was called (#5007).

Native SDK iOS

- Add new `ClearVREventTypes.AudioFocusChanged` event. The player will now receive callbacks when audio focus has been lost or regained (#4996).
- Fix issue where frame judder could be observed when playing content with a deep b-frames GOP structure (#5014).
- Fix issue where audio would be silenced for the rest of the session after being called. (#4996)
- Fix issue where the application would segfault in the rare cases that two subsequent video frames have identical RTS (#5003).
- Fix issue where
  `[ClearVR] Unable to convert Core.ProjectionType Unknown into appropriate Content format. Not implemented`
  warning would be printed when printing the ContentInfo of a ClearVR clip. This was a false positive warning (#5038).
- Fic issue where the ClearVRPlayer was unable to resume playback after a call was accepted while video was playing (#4998).

v8.1.1-1-g5a85a253 (2022-01-19)

Cross-platform

- Skip unsupported audio tracks where parsing mp4 file (#4927).
- Fix issue where `AudioTracksChanged` event was not triggered when switching audio tracks (#4939).
- The `VideoTrackInfo` in the `ContentInfo` object are now always sorted in descending order with the "highest quality" representation being first. See the docs for details (#4922).
- Fixed bug where player would not automatically exit paused state when reaching end of content and then seeking back (#4965).
- Fix bug where calling PopulateMediaInfo directly after CheckIsSupported on an HLS stream would incorrectly create a multi-feed HLS clip.
- Fixed issue where playback position and duration of VOD ClearVR content that doesn't start at segment 0 (e.g. Live2VOD) would be incorrect.
- Fixed issue where manifest-signalled bitrate was incorrect for manifest v3 content (#4981).

Unity SDK

- Fix issue where seek was called twice when dragging the seek bar in the Unity Sample Player project (#4925).
- Fix issue where rectilinear side-by-side stereoscopic content would play as if it were monoscopic (showing both left and right eye) after an app suspend->resume cycle.
- Android: add support for Widevine encrypted audio tracks (#4955).

v8.1-1-gbb6be1f6 (2021-12-20)

Cross-platform

- Add support for generic Widevine license servers
- Full rewrite of playback timing, fixing various issues with A/V sync in (long) live streams
- Added support for ABR in HLS
- Added support for independent audio tracks in HLS
- Added support for AAC 'packed' audio' in HLS
- Added support for AES-128 encryption in HLS
- Fixed several HLS-related corner cases
- Improved HLS Low Latency support
- Fixed various corner cases related to sync and seamless switching
- Updated Fish Eye rendering support, improving the perceived stereoscopic effect and overall picture quality
- Add `blackmagicUrsa12KCanon8158Mmf48K169` fish eye camera and lens preset
- `ClearVRDRMLicenseServerTypes.Unknown` enum case has been renamed to `ClearVRDRMLicenseServerTypes.Unspecified`.
- Deprecated `ClearVREventTypes.ABRLevelActivated`. This event type is not triggered anymore and references to it must be removed from your code. It has been replaced by `ClearVREventTypes.ActiveTracksChanged`, see below.
- Add `mediaInfo.getContentInfo()` API to fetch information (e.g. the resolution of available ABR representations, number of audio tracks, etc.) for the content being played.
- Add `ClearVReventTypes.ActiveTracksChanged` event that is fired when a (video/audio/subtitle) track has changed. For example, this is triggered after an ABR event.
- Fixed issue with audio looping if first segment in HLS VOD clip didn't have sequence number 0
- Fixed various causes for high memory pressure during repeatedly creating and destroying player contexts
- Fixed high memory consumption when joining HLS live streams that have been running for very long periods of time (months, years)
- Fixed corner case where ABR downswitch wouldn't go as smooth as possible in case segment length was not properly announced in manifest
- Fixed corner case where HLS ABR switch could result in higher than necessary bandwidth consumption
- Fixed bug that could lead to temporarily no audio after repeated ABR switches in HLS streams where audio is muxed in with video
- Fixed bug that could lead to slow switches and increased bandwidth consumption when

watching HLS streams close to the live edge while on a slow network
- Fixed bug where we could generate a bogus content-looped event during a live clip. This could also impact audio playback
- Fixed issue where ABR switches in HLS streams without EXT-X-PROGRAM-DATE-TIME tags and with inaccurate manifest-signalled segment lengths could result in jumps in time and audio issues
- Add support for specifying an ISO-639 language code to set the preferred audio track language. See `PrepareContentParameters()` and `SwitchContentParameters()` constructor for details.

Unity SDK

- The UnitySamplePlayer project has been upgraded from Unity `2018.4.27f1` to Unity `2020.3.1f1`.
- Add support for Universal Render Pipeline (URP) on Unity 2020+ (#3987).
- Fix issue where video rendering `GameObject` was positioned in the global coordinate system instead of the local coordinate system to the parent `GameObject`. Note that the original behaviour was *incorrect* and this might result in your mesh to be positioned in a different spot than it was previously (#4095).
- Fix issue where manually overriding the `DeviceType` by calling `Utils.RedetectDeviceType(DeviceType)` before player initialization was ignored during player initialization (#4069).
- Fix `NullReference` exception in `PopulateMediaInfoParameters.ToCoreProtobuf()` if the application was suspended before content preparation was started (e.g. very shortly after calling `clearVRPlayer.Initialize()`) (#4643).
- Fix `ClearVRPlayer.Intialize(Action<>)` callback not being triggered if application was suspended and resumed before content was fully loaded (#4643).
- iOS: fix rare automatic player stop when switching between two clips with the same video codec if both clips have b-frames (#4081).
- iOS: Unity has dropped support for OpenGL ES on Unity 2020.2.0f1 and newer (ref. https://unity3d.com/unity/whats-new/2020.2.0). This also means that the ClearVR SDK does not support it on these Unity versions.
- iOS and Android: fix crash when trying to calling controller.GetTimingReport() when no content was loaded (#4134).
- iOS and Android: add support for Google Cardboard XR Plugin. Requires Unity 2020.2+ (#3668).
- Android: improved support for Google Pixel 3 and 4 series phones. Previously, some higher resolution clips would fail to playback (#2361) and the video decoder would crash when switching between ClearVR and non-ClearVR content. A work-around for this driver issue has been added (#4173).
- iOS: No longer forcing swift version 4.0 for generated Xcode projects (#4718).
- Android: some video decoders can stall indefinitily under varying conditions (e.g. due to a corrupted bitstream or broken MediaCodec implementation by the vendor). A timeout has been added to make sure playback automatically stops when this happens (#4430).

- Android and PC: fix issue where cropped/padded pixels in a video bitstream (e.g. 1088p vs 1080p) could show up as green/garbled pixels on the edges of the video (#4169).
- PC: some libraries in `Assets/ClearVR/Plugins/x86_64` have been renamed. When importing this update you might be presented with a pop-up that will take care of removing these old libraries from your project (#4299, #4408).
- Android: fix audio sometimes disappears after `Seek()` or `SwitchContent()` on Pico Neo 3 devices (#4675).

Native SDK Android

- The ClearVR Sample Player project has been updated.
- Add playback of rectilinear ("16x9") video playback.
- Orientation tracker management has changed. `clearVRPlayer.attachClearVROrientationTracker()` and `clearVRPlayer.detachClearVROrientationTracker()` have been deprecated and cannot be called anymore. By default, the `ClearVRPlayer` will now enable Touch only control for you automatically as soon as it detects that you are playing omnidirectional (180/360) video. If you are playing rectilinear content, Touch control will be disabled automatically.
- The `ClearVRPlayerParameters()` constructor has been simplified, taking only the required arguments. You can set individual fields on your object as needed after construction but before calling `clearVRPlayer.initialize()`.
- improved support for Google Pixel 3 and 4 series phones. Previously, some higher resolution clips would fail to playback (#2361) and the video decoder would crash when switching between ClearVR and non-ClearVR content. A work-around for this driver issue has been added (#4173).
- Fix issue where cropped/padded pixels in a video bitstream (e.g. 1088p vs 1080p) could show up as green/garbled pixels on the edges of the video (#4169).
- Some video decoders can stall indefinitily under varying conditions (e.g. due to a corrupted bitstream or broken MediaCodec implementation by the vendor). A timeout has been added to make sure playback automatically stops when this happens (#4430).
- Fix issue where calling `clearVRPlayer.initialize()` in your Activity's `onCreate()` or `onStart()` would cause a `Cannot set SetClearVRCoreWrapperObject multiple times` crash (#4347).
- Support for the GearVR framework is removed (#4822).

Native SDK iOS

- Add AVC support.
- Add playback of rectilinear ("16x9") video playback.
- Getter methods for class prorperties have been deprecated and replaced by computed (getter) properties instead. Existing getter APIs will be removed after 2022-06-30.
- Orientation tracker management has changed. `clearVRPlayer.attachClearVROrientationTracker()` and `clearVRPlayer.detachClearVROrientationTracker()` have been deprecated and cannot be called anymore. By default, the `ClearVRPlayer` will now enable Touch only control for you automatically as soon as it detects that you are playing omnidirectional (180/360) video. If you are playing

rectilinear content, Touch control will be disabled automatically.
- The `TMClearVRPlayerParameters()` constructor has been simplified, taking only the required arguments. You can set individual fields on your object as needed after construction but before calling `clearVRPlayer.initialize()`.
- Fix rare automatic player stop when switching between two clips with the same video codec if both clips have b-frames (#4081).

Known issues

- Unity: SRP batching is not supported on Android due to bugs in Unity's GLSL shader compiler. When SRP batching is enabled, the mesh will always be positioned at (0, 0, 0) with scale (1, 1, 1) regardless of the Mesh' Transform (model matrix). SRP batching will be automatically disabled by the SDK under these conditions (#4195).

v8.0.15-7-g837e8754 (2021-12-03)

Unity SDK

- iOS: `ApplicationFocusAndPauseHandlingTypes.Legacy` cannot be used on this platform. If selected, `ApplicationFocusAndPauseHandlingTypes.Recommended` will be forced instead (#4862).

Native SDK iOS

- `TMApplicationFocusAndPauseHandlingTypes.Legacy` cannot be used on this platform. If selected, `TMApplicationFocusAndPauseHandlingTypes.Recommended` will be forced instead (#4862).

v8.0.14-1-g1d93f8db (2021-11-10)

Unity SDK

- Android: fix issue where enabling aggressive compiler optimizations in Unity would result in some symbols being stripped from compiled code that are required by the ClearVR SDK (#4818).

v8.0.13-1-g8187e3f9 (2021-09-20)

Unity SDK

- iOS: fix player crash if a third party library exposes C-style APIs that have an identical function signature as C-style APIs exposed by ClearVR's `MediaFlow.framework` (#4691).

v8.0.12-1-gc6ee9385 (2021-08-31)

Unity SDK

- iOS: add work-around for `fatal error: memory reservation exceeds address space limit` crash observed on iPhone 12 model devices. This is a problem in a third-party dependency, see

https://github.com/golang/go/issues/46860.

Native SDK iOS

- Add work-around for `fatal error: memory reservation exceeds address space limit` crash observed on iPhone 12 model devices. This is a problem in a third-party dependency, see https://github.com/golang/go/issues/46860.

v8.0.11-1-g57bce0d1 (2021-08-26)

Unity SDK

- Editor: fix Unity Editor crash when your enter playmode, start a ClearVRPlayer, stop it, start another ClearVRPlayer, leave playmode without stopping the ClearVRPlayer, enter playmode and start another ClearVRPlayer (#4496).
- Editor: fix crash when switching from Direct3D11 to OpenGLCore graphics API after at least one clip was played (#4400).
- PC: fix rare crash in protobuf library when restarting playback after failed initialization (#4425).
- PC: fix random crash when setting parameters fails (#4365).
- Fix crash if stopping playback after content was loaded but before a Mesh was allocated (#4579).

v8.0.10-1-ga6b5b6b0 (2021-07-08)

Unity SDK

- Android: querying DRM playback capabilities on the Pico Neo 3 results in an infinite hang. As a work-around, DRM support has been disabled on this device until this bug has been fixed by PicoVR (#4388).

Native SDK Android

- Querying DRM playback capabilities on the Pico Neo 3 results in an infinite hang. As a work-around, DRM support has been disabled on this device until this bug has been fixed by PicoVR (#4388).

v8.0.9-1-g1db6f3d6 (2021-07-03)

Unity SDK

- `ClearVRPlayer.DestroyMediaPlayer()` is a private API now and cannot be called from the application layer.
- Android and iOS: fix
  `Application did not properly unload previous player instance. Forcing clean-up, but this is not officially supported!`
  error message on console when application regains focus. The was a false-positive warning message (#4381).

- iOS: fix issue where playback would not resume when the notification center was pulled down and up again (#4380).
- Android: fix very rare crash in audio decoder when stopping playback (#4361).

Native SDK Android

- Fix very rare crash in audio decoder when stopping playback (#4361).

v8.0.8-1-gc5cb3ca0 (2021-06-04)

Unity SDK

- Editor: fix issue where the Unity Editor would sometimes quit play mode when a ClearVRPlayer object was stopped (#4259).
- PC: fix issue where `GetTimingReport()` returned seemingly random values (#4256).
- PC: fix rare crash when calling `GetCurrentContentTimeInMilliseconds()` and/or `GetContentDurationInMilliseconds()` during ClearVRPlayer initialization (#2276, #4274).
- iOS: fix random crashes due to bug in third party dependency (#3881)

Native SDK iOS

- Fix random crashes due to bug in third party dependency (#3881)

v8.0.7-7-gade7fabb (2021-05-31)

Cross-platform

- Fixed issue where selecting an initial start position far into a (long) VOD HLS clip could lead to the player refusing playback and throwing a seemingly unrelated error (#4239).

Unity SDK

- Fix issue where creating a new ClearVRPlayer in `StateChangedStopped` or the `Stop()` callback would result in an `Application did not properly unload previous player instance. Forcing clean-up, but this is not officially supported` message on the console. This was a false-positive warning as this application logic is supported by the SDK (#4228).
- Android: fix issue where Samsung S21 series devices would suffer a hard reset when switching to/from specific 12K and 16K content. This crash is caused by a bug in the video decoder driver and a work-around has been added (#4224).
- PC: fix issue where `GetTimingReport()` returned a `TimingReport` object that always had all its fields (e.g. position, duration and seekable range) set to 0 (#4227).
- PC: fix issue where a stand-alone application could hang when closed using ALT+F4 (#4244).

Native SDK Android

- Fix issue where Samsung S21 series devices would suffer a hard reset when switching to/from specific 12K and 16K content. This crash is caused by a bug in the video decoder driver and a work-around has been added (#4224).

v8.0.6-1-ga20b6424 (2021-05-19)

Cross-platform

- Fix issue where playback would stop when attempting to `SwitchContent()` to a ClearVR clip for the second time of which the `manifest.json` is still available on the CDN but the segments are not (#4201).

Unity SDK

- iOS: improve native plugin management. In previous versions, `libClearVRNativeRendererPlugin-XXXX.a` plugins were dynamically renamed to `.disabled` depending on what Unity version you are targetting. Now, they will be enabled/disabled through Unity's plugin management interface. Please note that when upgrading to this SDK or newer, any `libClearVRNativeRendererPlugin-XXXX.a.disabled` plugin in your repository will be automatically renamed to `libClearVRNativeRendererPlugin-XXXX.a`. Also, your build will fail if you leave any `libClearVRNativeRendererPlugin-XXXX.a.disabled` files around in your repository (#2830).
- iOS: add Unity 2020.2+ compatible native plugin `libClearVRNativeRendererPlugin-2020.2.a`. Note that `libClearVRNativeRendererPlugin-2018.a` will be used for Unity 2018 - 2020.1 (#4161).
- iOS: XCode 12.5 support (#4199).

Native SDK iOS

- XCode 12.5 support (#4199).

v8.0.5-1-g435b8e4f (2021-05-10)

Cross-platform

- Fix issue where the playout could freeze if a broken URL was fed into a `SwitchContent()` call.
- Do not allow ABR upswitch when playing 3D content if the upswitch would require 2D playback to fit the decoder constraints.

Unity SDK

- Add new `RenderModes.ForcedMonoscopic`. When set to this mode, content playback will always stick to monoscopic whatever the clip you `SwitchContent()` to.
- Fix `Utils.GetType()` failing when iterating over all assemblies as a last attempt when looking for a `Type` (#4171).
- Android: content playback would fail if a `ContentItem` specified non-null `DRMInfo` but the clip was not DRM protected (#4066).

Native SDK Android

- Content playback would fail if a `ContentItem` specified non-null `DRMInfo` but the clip was not DRM protected (#4066).

v8.0.4-1-g920745e6 (2021-04-21)

Cross-platform

- Fix crash if `allowDecoderContraintsInducedStereoToMono = true` and stereoscopic clip does not fit in decoder while running on an HMD (#4013).
- Improved stereo-as-mono playback on decoder constrained devices (#4140).

Unity SDK

- Fix issue where the `GameObject` holding the `ClearVRMesh` was not destroyed if it was not attached to the `GameObject` that holds the `ClearVRPlayer` object (this could happen if `platformOptions.parentGameObject` was explicitly set) (#3997).
- Fix crash when closing the Unity Editor or a standalone application using ALT+F4 during video playback (#3928).
- iOS and PC: fix issue where a `ContentItem`'s custom (non-default) `startViewportAndDisplayObjectPose` was ignored when using the `SwitchContent` API (#4016).
- iOS and PC: fix issue where a Seek or SwitchContent with `TimingTypes.LiveEdge` would yield undefined behaviour if targetPosition was set to a value != 0. This was unlikely to happen under normal conditions and this did not affect Android.

v8.0.3-1-ge4d40582 (2021-03-12)

Cross-platform

- Fix issue with track ordering in progressive MP4 content.
- Increase maximum supported size of progressive MP4 'moov' box.
- Fix issue where last segment in HLS VOD segment would not always be played out.

Unity SDK

- Fix issue where a custom `ViewportAndDisplayObjectPose` on the `ContentItem` was not effectuated when loading it as the first clip (#3967).
- Android: fix `SwitchContent` sometimes failing if no `TimingParameters` were specified.
- Android: fix issue where decoder capabilities were not always properly parsed on devices that support Tunneled playback (#3955).
- Android: fix issue where
  `The required video mimetype '<mimetype>', codec profile '<profile>' and decoder flags '<flags>' is supported by this SDK, but no hardware or software decoder was found.`
  was thrown while loading a `ContentItem` even if the device supported the specified combination just fine (#3971).

Native SDK Android

- Fix `SwitchContent` sometimes failing if no `TimingParameters` were specified.
- Fix issue where decoder capabilities were not always properly parsed on devices that support Tunneled playback (#3955).
- Fix rare `java.lang.IllegalArgumentException: callback must not be null` exception in `FrameReleaseChoreographer` (#3917).
- Fix issue where `The required video mimetype '<mimetype>', codec profile '<profile>' and decoder flags '<flags>' is supported by this SDK, but no hardware or software decoder was found.` was thrown while loading a `ContentItem` even if the device supported the specified combination just fine (#3971).

v8.0.2-1-gfd1eb447 (2021-03-05)

Cross-platform

- Fix possible A/V desynchronisation in some clips that contain B-frames (#3919)
- Fix issue in query string parsing in HLS master manifest path (#3932)

Unity SDK

- Android: fix issue where sometimes 1 to 4 frames were briefly shown after a seek or switch content from the point before the seek or switch content when the clip contains B-frames (#3922).

Native SDK Android

- Fix issue where sometimes 1 to 4 frames were briefly shown after a seek or switch content from the point before the seek or switch content when the clip contains B-frames (#3922).
- Fix 180 degree content not rendering correctly (#3935)

v8.0.1-4-g5716b608 (2021-02-26)

Cross-platform

- Add support for >180 degree FishEye content. The mesh is now a full 360 degree ERP sphere that is transparent for the parts that are not covered by the original video.

Unity SDK

- Android: fix DeleteGlobalRef-crash in JNIBridge when setting `ContentItem.FishEyeSetting != null`.
- Android: fix audio playback failed on some devices with the message: `Unable to set audiotrack back to play.` (#3902).
- iOS: fix playing a `.tmm` file would result in `all items in meta-manifest are unsupported` error message (#3904).

Native SDK Android

- Fix audio playback failed on some devices with the message: `Unable to set audiotrack back to play.` (#3902).

Native SDK iOS

- Fix playing a `.tmm` file would result in `all items in meta-manifest are unsupported` error message (#3904).

v8.0-20-g2b064e8b (2021-02-21)

Unity SDK

- Remove ambiguous deprecated `ContentItem` constructor.
- Android: fix issue where playback on LG V505 paired with Galaxy Buds bluetooth headset resulted in stuttering audio.

Native SDK Android

- Fix issue where playback on LG V505 paired with Galaxy Buds bluetooth headset resulted in stuttering audio.

v8.0-1-g2ad6cd87 (2021-02-18)

For details on upgrading to v8, please refer toupgrading to v8

New features

- Unity editor support (Windows, Linux)
- Add support for seamless switch content.
- Add support for inter-device synchronized playback in live events.
- Add support for playback of VOD clips as a live event.
- Add basic support for adaptive bitrate with ClearVR content.
- Improved HLS playback support

Cross-platform

- Add support for Seamless SwitchContent.
- Initial support for adaptive bitrate.
- `playback.enable_automatic_abr` has been renamed to `advanced.abr.enable`
- Automatic fallback from 3D to 2D in poor network conditions wher`playback.enable_automatic_abr` was enabled has been removed. There is no similar mechanism currently.
- Fix ABR performance statistics incorrectly resetting under specific conditions.
- Fixed cache hit ratio metric not being correct when using Cloudfront CDN.
- Set cache hit ratio metric to -1 if cache hit ratio cannot be determined (before this would result in a cache hit ratio of 100%).

- Re-use open TCP connnections when switching content (or switching ABR levels) to content hosted on the same domain, reducing SwitchContent() latency.
- Fix: issue with ABR where unsupported ABR levels would be attempted multiple times.
- Allow playing stereoscopic content monoscopically if decoder level doesn't support playing back stereo
- Fix: a very rare bug in the tmrange package that could case a fatal error in the SDK (#3772)
- Fix: bug where wrong requests could be cancelled during an ABR switch, causing the player to get stuck in buffering indefinitely

Unity SDK

- New UnitySamplePlayer project
- Windows and Linux: Unity Editor support
- The SDK is now .NET 4.x compatible.
- iOS: the SDK now supports bitcode.
- Add support for changing screen orientation while video is playing. Previously, fallback tiles might be perceived at the edges of the screen.
- Add `platformOptions.enableABR` and `platformOptions.abrStartMode` fields to control the experimental ABR algorithm (disabled by default).
- `platformOptions.clearVRCoreVerbosityLevel` and `platformOptions.clearVRCoreLogToFile` have been replaced by `static ClearVRPlayer.coreLogLevel` and `static ClearVRPlayer.coreLogFile` respectively.
- `ClearVREventTypes.ABRSwitch` has been renamed to `ClearVREventTypes.ABRLevelActivated`.
- Add support for specifying an initial audio gain.
- ClearVREventTypes.ContentFormatChanged, .RenderModeChanged and .FirstFrameRendered are now guaranteed to be called in this order. Previously, the order was undefined.
- Within one vsync, multiple ClearVREvents may now be received. Previously, they were triggered one-by-one per vsync.
- Fix: maintain audio gain and mute state when suspending and resuming the application.
- Android and Windows: add support for changing the playback rate on the fly.
- Android: fix issue where audio playback over bluetooth headset could result in stuttering.
- Add advanced ContentSupportedTesterParameters.allowDecoderContraintsInducedStereoToMono and platformOptions.allowDecoderContraintsInducedStereoToMono field. This advanced parameter can be enabled to allow the video player to automatically play stereoscopic content as monoscopic if the video decoder has insufficient capacity to playback the full stereoscopic video. By default, this is disabled.
- Android: a proguard txt file has been added to the SDK, that prevents the `com.tiledmedia.**` namespace from being touched by proguard.
- Android: fix issue where audio/video desynchronisation could be observed on certain device/(bluetooth) headset combinations.

- Add clearVRPlayerParameters.enableABR and clearVRPlayerParameters.abrStartMode fields to control the experimental ABR algorithm (disabled by default).
- Include x86 and x86_64 versions of the SDK to support building your app for the Android emulator. Note that the SDK does NOT officially support playback in the emulator, so your milage may vary.
- Fix: maintain audio gain and mute state when suspending and resuming the application.
- Fix: issue where audio playback over bluetooth headset could result in stuttering.
- `clearVRPlayerParameters.clearVRCoreVerbosityLevel` and `clearVRPlayerParameters.clearVRCoreLogToFile` have been replaced by `static ClearVRPlayer.coreLogLevel` and `static ClearVRPlayer.coreLogFile` respectively.
- `ClearVREventTypes.ABRSwitch` has been renamed to `ClearVREventTypes.ABRLevelActivated`.
- Add advanced ContentSupportedTesterParameters.allowDecoderContraintsInducedStereoToMono and clearVRPlayerParameters.allowDecoderContraintsInducedStereoToMono field. This advanced parameter can be enabled to allow the video player to automatically play stereoscopic content as monoscopic if the video decoder has insufficient capacity to playback the full stereoscopic video. By default, this is disabled.
- Fix: issue where audio playback over bluetooth headset could result in stuttering.
- Fix: issue where audio/video desynchronisation could be observed on certain device/(bluetooth) headset combinations.

Native SDK iOS

- The SDK now supports bitcode.
- Fix: properly maintain pan speed when resizing view. Previously, pan speed was based on the initial dimensions of the view.
- Add clearVRPlayerParameters.enableABR and clearVRPlayerParameters.abrStartMode fields to control the experimental ABR algorithm (disabled by default).
- Fix: maintain audio gain and mute state when suspending and resuming the application.
- `clearVRPlayerParameters.clearVRCoreVerbosityLevel` and `clearVRPlayerParameters.clearVRCoreLogToFile` have been replaced by `static TMClearVRPlayer.coreLogLevel` and `static TMClearVRPlayer.coreLogFile` respectively.
- `TMClearVREventTypes.ABRSwitch` has been renamed to `TMClearVREventTypes.ABRLevelActivated`.
- Add advanced ContentSupportedTesterParameters.allowDecoderContraintsInducedStereoToMono and clearVRPlayerParameters.allowDecoderContraintsInducedStereoToMono field. This advanced parameter can be enabled to allow the video player to automatically play stereoscopic content as monoscopic if the video decoder has insufficient capacity to playback the full stereoscopic video. By default, this is disabled.

v7.4.9-1-gd14953ac (2021-02-17)

Unity SDK

- Improve automatic device detection in Unity projects that use packages with splitted assemblies.

Native SDK Android

- Fix issue where your `clearVRPlayer.initialize()` callback was not always triggered if `clearVRPlayer.stop()` was called at a specific moment in time during initialization.
- Fix issue where `ClearVRPlayer.testIsContentSupporteD()` was not cancelled if `clearVRPlayer.stop()` was called but `clearVRPlayer.initialize()` was not (yet) called.

Native SDK iOS

- Fix `ClearVREventTypes.FirstFrameRendered` not being emitted when switching between clips with different `ContentFormats`.

v7.4.8-1-g92350536 (2021-02-02)

Native SDK Android

- Fix "Attempt to remove non-JNI local reference" warning message in logcat (#3771).
- Fix rare NullReference exception in ClearVRTextureView.RenderThread class (#3751, #3763).

Native SDK iOS

- Fix issue where first couple of frames were sometimes not rendered to the screen (dropped) when playing the first clip (#3711).

v7.4.7-1-g9e79c492 (2021-01-08)

- Fixed bug where destroying player while DRM was being initialized could result in call taking longer than necessary.

Unity SDK

- Android: fix issue where the `JNIBridges.Setup()` method could sometimes get stripped away in case aggresive compiler optimizations are enabled in Unity.
- iOS: reduced peak memory pressure when rapidly switching content.

Native SDK iOS

- Add support for playback of equirectangular 180/360 and fish eye (equisolid and equidistant) content in HLS and progressive-MP4 file format.
- Reduced peak memory pressure when rapidly switching content.

v7.4.6-1-g56aa0aec (2020-12-21)

Unity SDK

- Android: fix stereoscopic video being rendered as monoscopic on PicoVR devices if the camera rig is misconfigured (specifically: if the `LeftEye` component is incorrectly tagged as `MainCamera`).
- Android: fix playback not always starting on certain Android device. This issue was introduced in v7.4.5-1-g8371cc1a.

Native SDK Android

- Fix playback not always starting on certain Android device. This issue was introduced in v7.4.5-1-g8371cc1a.

v7.4.5-1-g8371cc1a (2020-12-17)

This version was pulled because of a bug in the Android stack. This issue was fixed in v7.4.6-1-g56aa0aec

Cross-platform

- Fix issue where some stereoscopic clips could not be played on some 4K decoders, despite having sufficient decoder capacity.

Unity SDK

- Android: fix issue where DRM-resources were not always properly released when switching between DRM-protected clips using different encryption ciphers.
- Android: improve audio playback performance
- iOS: fix rare crash in audio decoder stack when calling `SetAudioGain()`

Native SDK Android

- Add support for playback of equirectangular 180/360 and fish eye (equisolid and equidistant) content in HLS and progressive-MP4 file format.
- Improve audio playback performance.
- Fix rare null reference exception in `clearVRPlayer.selectNonSpatialAudioTrack()`.

Native SDK iOS

- Fix rare crash in audio decoder stack when calling `setAudioGain()`

v7.4.4-1-gf63c934b (2020-11-27)

Cross-platform

- Fix issue where long (1 hour +) progressive-mp4 clips could not be played.

Unity SDK

- iOS: fix issue where interaction requests (e.g. seek, switch content, stop) were handled serially instead of concurrently. This significantly improves player responsiveness when interaction requests are fired rapidly.

Native SDK iOS

- Fix issue where interaction requests (e.g. seek, switch content, stop) were handled serially instead of concurrently. This significantly improves player responsiveness when interaction requests are fired rapidly.

Native SDK Android

- Fix NullReference exception in EGLRenderThread when rotating the ClearVRTextureView component before initializing the ClearVRPlayer component.

v7.4.3-1-g6df303c7 (2020-11-16)

Unity SDK

- iOS: libraries are now built against XCode 12.2 (12B45b)

Native SDK iOS

- Libraries are now built against XCode 12.2 (12B45b)

v7.4.2-11-gc081191e (2020-11-13)

Cross-platform

- Fix rare crash during TestIsContentSupported.

Unity SDK

- Fix: the http user agent could only be set once. Any subsequent changes were ignored.

v7.4.2-1-gb51aeb0c (2020-11-03)

Cross-platform

- Fix issue where micro buffering could be observed (typically as audible ticks or glitches) when a live stream was playing very close to the live edge.
- Fix issue where the application could go out of memory when switching between many HLS clips.
- Fix issue where the player could get stuck if stopped at a specific moment in time while loading a DRM-protected clip.
- Fix issue where the player could get stuck in buffering when switching between HLS clips.

Unity SDK

- Android: fix possible crash when switching rapidly between streams that are DRM-protected with different keys.
- Fix issue where the player could get stuck if stopped at a specific moment in time while loading a DRM clip.

v7.4.1-1-ga4287bc6 (2020-10-29)

Cross-platform

- Significantly improve overall performance of DRM-protected content playback.

Unity SDK

- iOS: fix issue where playback of a clip without any audio track resulted in an error.

Native SDK iOS

- Fix issue where playback of a clip without any audio track resulted in an error.

v7.4-1-gf770b4e2 (2020-10-22)

Cross-platform

- Content supported check is now up to twice as fast.

Unity SDK

- TestIsContentSupported callback is now guaranteed to be triggered on the main Unity thread if the API was called from the main Unity thread.
- Add support for arbitrary FishEye clips. You can now tune the exact lens parameters by specifying the `FishEyeSettings` when constructing a `ContentItem`.
- Add support for playback of rectilinear (16:9) stereoscopic top-bottom video clips.
- Android: remove dependency on `support-compat-25.3.1.aar`. The ClearVR SDK does no longer depend on `AppCompat` nor on `Androidx` support libraries.
- Android and iOS: fix stereoscopic playback on Mobfish cardboard SDK.
- Windows: fix rare crash that could be triggered by panning around while calling pause/unpause in as specific pattern.

Native SDK Android

- Remove dependency on `support-compat-25.3.1.aar`. The ClearVR SDK does no longer depend on `AppCompat` nor on `Androidx` support libraries.

v7.3.5-4-g66b1e682 (2020-10-21)

Unity SDK

- Main Unity thread sometimes got briefly blocked after completing a switch content request,

resulting in temporarily degraded application performance.
- Android: automatically detect center/head camera on PicoVR SDK 2.8.6_B488-20200807.
- Android: fix issue where you could not always switch from a non-DRM protected clip to a DRM protected clip.

Known issues

- Android: enabling PicoVR's `UseSinglePass` option results in spurious `GL_INVALID_OPERATION: Operation illegal in current state` warning message on logcat. The message is harmess, but a work-around is to disable `UseSinglePass` feature until this issue is resolved.

v7.3.4-1-g25b9517b (2020-10-19)

Cross-platform

- Fix: playback would stop unexpectedly if pMP4 clip did not contain any audio track.

Unity SDK

- No changes

Native SDK iOS

- Fix: allow TMApplicationFocusAndPauseHandlingTypes.Disabled and .Legacy modes to be selected.
- Fix: player could get stuck in stopping when interrupting initial player setup twice by pushing application to the background.
- Fix: VOD playback would not restart from the last known content position after application regained focus when pulling the notification center down. It started from the beginning instead. This did not happen when pushing the application to the background using the home button.
- Fix: rare crash in ResignActiveNotification handler when pushing application to the background *just* after preparing the first clip for playback.
- Fix: `clearVRPlayer.initialize()` completion handler was sometimes not triggered if initialization was interrupted by pushing the application to the background and bringing it back to the foreground.
- Fix: `clearVRPlayer.stop()` closure not always being triggered if called shortly after calling `clearVRPlayer.initialize()`.
- Fix: no `StateChangedStopped` was received if `clearVRPlayer` object was stopped before `StateChangedPreparingCore` was reached.

Native SDK Android

- Fix: `clearVRPlayer.initialize()` completion handler was sometimes not triggered if initialization was interrupted by pushing the application to the background and bringing it back to the foreground.

v7.3.3-1-ga1cc4019 (2020-10-12)

Cross-platform

- Fix audio suddenly disappearing when seeking multiple times in pMP4 clip.

Unity SDK

- Add `ContentItem.approximateDistanceFromLiveEdgeInMilliseconds` field to override minimum distance from live edge. Read the documentation, as changing from its default value (0) is strongly discouraged.
- iOS: shuffled tiles sometimes being visible in case of high system load (bug introduced in v7.3.2-4-gadb2856e (2020-09-30))
- iOS: application could go out of memory if application framerate was lower than video framerate for prolonged period of time.
- iOS: if a clip contained only (an) unsupported audio track(s), it could not be played.

Native SDK Android

- Fix: touch control being broken when application is locked in landscape orientation but device is held in portrait.
- Fix: gracefully handle OpenGLES error instead of throwing a `RuntimeException()`.
- Changed: touch now always follows your finger, also when zoomed in.
- Changed: increased default maximum zoom-in level.
- Add `ContentItem.approximateDistanceFromLiveEdgeInMilliseconds` field to override minimum distance from live edge. Read the documentation, as changing from its default value (0) is strongly discouraged.

Native SDK iOS

- Fix: shuffled tiles were sometimes observed in case of high system load (bug introduced in v7.3.2-4-gadb2856e (2020-09-30))
- Fix: touch control being broken when application is locked in landscape orientation but device is held in portrait.
- Fix: race condition when stopping playback while media info is parsed.
- Fix: race condition in audioDispatchQueue handler.
- Fix: app lifecycle tracking not always being properly detached, causing a crash on ResignActiveNotification handler.
- Fix: rare crash which can be traced back to `CVR_NRP_CreateMeshRenderer()`
- Fix: application could go out of memory if application framerate was lower than video framerate for prolonged period of time.
- Fix: if a clip contained only (an) unsupported audio track(s), it could not be played.
- Changed: touch now always follows your finger, also when zoomed in.
- Changed: increased default maximum zoom-in level.
- Add `TMContentItem.approximateDistanceFromLiveEdgeInMilliseconds` field to override minimum distance from live edge. Read the documentation, as changing from its default value (0) is

strongly discouraged.

v7.3.2-4-gadb2856e (2020-09-30)

Cross-platform

- Fix memory leak in HLS and pMP4 playback when playing for a long time.
- Fix occasional memory leak in ClearVR playback.
- Fix rare lock-up when switching audio track while switching between content items.
- Improve HTTP/TCP session reusage after error.
- Improve handling of video decoder overflow if device cannot keep up with source framerate.
- pMP4: significantly decrease network load when seeking and reduce seek-time.

Known issues

- pMP4: audio will glitch/"disappear" when seeking repeatedly in pMP4 clips.

Unity SDK

- Add `MediaPlayer.UpdateOverrideUserAgent()` API to change the HTTP user agent during playback. Usage of this API is highly discouraged, please read the in-line documentation for details.
- iOS: Fix lockup/crash to homescreen when switching content.
- iOS: Fix typo in error message when attempting to use OpenGLES 3 + Multithreaded rendering on Unity 2019_2_OR_NEWER
- iOS: Fix issue where no audio is played back when testIsContentSupported() is called prior to loading a clip.
- iOS: Fix audio gain resetting after each switch content.
- Android: Remove `android:allowBackup=true` from library's AndroidManifest.xml. Improves compatibility with newer versions of Oculus Utilities for Unity.
- Android and iOS: add support for the Mobfish cardboard SDK.

Known issues

- Android and iOS: stereoscopic videos are rendered monoscopic when using the Mobfish SDK.

Native SDK Android

- Add pinch to zoom to Touch and Motion+Touch orientation tracker
- Fix rare crash when destroying view just after creating it.
- Fix `ClearVRPlayer` getting stuck in stopping state as a result of a specific network error.
- Remove dependency on `support-compat-25.3.1.aar`. We are now compatible with the legacy `AppCompat` as well as `Androidx` Android libraries.

Known issues

- Motion+Touch controller does not properly track Touch when locking app in landscape and

holding device in portrait.

Native SDK iOS

- Add pinch to zoom to Touch and Motion+Touch orientation tracker
- Fix lockup/crash to homescreen when switching content.
- Fix typo in error message when attempting to use OpenGLES 3 + Multithreaded rendering on Unity 2019_2_OR_NEWER
- Fix issue where no audio is played back when testIsContentSupported() is called prior to loading a clip.
- Fix audio gain resetting after each switch content.
- Change ClearVRUIView's background from white to transparent, allowing you to easily modify the default background color.
- Fix testIsContentSupported() completion handler being triggered on a random background thread instead of the main UI thread.
- Fix `ClearVRPlayer` getting stuck in stopping state as a result of a specific network error.

Known issues

- Motion+Touch controller does not properly track Touch when locking app in landscape and holding device in portrait.

v7.3.1-1-g8e4ae86a (2020-08-21)

Cross-platform

- Add support for chained cookies in manifest URI.
- Add support for ACTS segmented streams.
- Fix rare crash when stopping playback while loading content.

Unity SDK

- Fix issue where seeking to the live edge using SeekFlags.LiveEdge was parsed as an illegal seek request.

Native SDK Android

- Fix missing check if trying to initialize a ClearVRPlayer while another one is (still) active. As a result, a crash could occur.
- Fix issue where seeking to the live edge using SeekFlags.LiveEdge was parsed as an illegal seek request.
- Deprecate SeekParameters(long) constructor as it was easily confused with SeekParameters(long, long) which could result in unexpected seek behaviour. The latter is now the preferred way of seeking. The former API will be removed after 2020/12/31. One is highly encouraged to migrate immediately.

Native SDK iOS

- Fix missing check if trying to initialize a ClearVRPlayer while another one is (still) active. As a result, a crash could occur.
- Fix issue where seeking to the live edge using SeekFlags.LiveEdge was parsed as an illegal seek request.

v7.3-1-g09a8bb1c (2020-08-13)

Cross-platform

- Fix edges of viewport not always being covered by high-quality tiles when playing on a non-headset (flat) device.

Unity SDK

- Updated readme and API documentation. PDFs can be found under Assets/ClearVR/Docs, an online version can be accessed through the Unity Editor ClearVR -> About menu item.
- BREAKING: ClearVRMessageCodes.GenericOK has been marked deprecated and cannot be used. One should replace GenericOK by ClearVRCoreWrapperGenericOK
- BREAKING: ClearVREvent.clearVRAsyncRequestResponse is no longer accessible. To get access to your optional arguments, please use ClearVREvent.optionalArguments instead.
- Removed InitializeFlags.LongBuffer flag, it was removed a while ago.
- Fix crash when initializing a new ClearVRPlayer in the `StateChangedStopped` event.
- Add `String platformOptions.overrideUserAgent` field to override the HTTP user agent inserted in each HTTP request. Please read the inline documentation in case you consider using it, as using it can result in severe network performance degradation.
- Add `TimingReport controller.GetTimingReport()` API, allowing you to get detailed timing-related information, including seek boundaries. Note that this API is only available on Android and iOS, and will return dummy values on PC.
- Android: improved support for switching between DRM and clear content. You can now freely SwitchContent between Widevine L3 encrypted content and clear content without the need of destroying and creating a ClearVRPlayer object.
- Android: add support for AVC video playback (currently only supported for progressive MP4, HLS to be added soon).
- Android: add support for automatically detecting HEVC video decoder capabilities, enabling 8K decoder support on any capable device.
- Android: fix crash if trying to SwitchContent while the first clip is still loading.
- Android and iOS: add `static ClearVRPlayer.TestIsContentSupported(Params, Action<>)` API to test whether a ContentItem is supported or not.
- Android and iOS: add `contentSupportedStatus` field to `ContentItem` which signals the supported status of the `ContentItem` . By default, it is set to `.Unknown`, after testing it can be any of the `ContentSupportedStatus` enum values: `.Unknown` (could not be tested, e.g. codec not supported for testing), `.Unsupported` (device cannot play the content) and `.Supported` (content is tested and

supported).
- iOS: fix crash when stopping playback while ClearVRPlayer is loading the first clip.
- PC: improved pMP4 and HLS support.
- PC: various bugfixes related to audio playback.

Native SDK Android

- BREAKING: `ClearVRPlayerParameters()` constructor now takes a reference to your Activity, rather than the `ApplicationContext`, as third argument.
- Fix player now properly releasing all resources when the app looses focus. The player will restart playback as soon as the app regains focus.
- Fix viewport not always covering full view when rotating the view's orientation.
- Removed InitializeFlags.LongBuffer flag, it was removed for a while ago.
- Fix magic window mode being off by about 135 degrees to the left.
- Fix magic window mode not taking device roll into account.
- Improve touch mode around the poles. You can no longer cross the pole when in touch mode. Note that this is still possible when in the combined magic window + touch mode.
- Add `static ClearVRPlayer.testIsContentSupported(Params, Callback)` API to test whether a `ContentItem` is supported or not.
- Performance and stability improvements.
- Add `TimingReport controller.GetTimingReport()` API, allowing you to get detailed timing-related information, including seek boundaries.

Native SDK iOS

- Fix potential crash when hammering start -> stop -> start.
- Fix magic window mode being off by about 45 degrees to the right.
- Fix magic window mode not taking device roll into account.
- Add combined touch + magic window mode.
- Improve touch mode around the poles. You can no longer cross the pole when in touch mode. Note that this is still possible when in the combined magic window + touch mode.
- Expose more SDK features in the Sample Player.
- Fix magic window mode not working correctly in portrait and portraitUpsideDown.
- Fix crash when stopping playback while ClearVRPlayer is loading the first clip.
- Add `static TMClearVRPlayer.testIsContentSupported(Params, Completion)` API to test whether a `TMContentItem` is supported or not.
- Performance and stability improvements.
- Add `TimingReport controller.GetTimingReport()` API, allowing you to get detailed timing-related information, including seek boundaries.

v7.2.3-3-g2b8a2501 (2020-05-25)

Unity SDK

- Fix viewport resetting after each switch content on flat (non-headset) devices, even if the mesh type did not change.
- PC: increase incorrect initialize timeout (5 seconds) to the correct default value of 30 seconds.

v7.2.2-1-g6368f867 (2020-05-11)

Cross-platform

- Support self-signed certificates

Unity SDK

- PC: Fixed various crashes related to audio playback on Windows and Linux.

Native SDK iOS

- Fix CAMetalLayer not being resized when the ClearVRUIView is resized.
- The library can now be linked against iOS 9 and up, yet only iOS 11 and up are supported on run-time.

Native SDK Android

- Fix video being flipped upside-down when changing the device's orientation.

v7.2.1-5-g7b1149d4 (2020-04-22)

Unity SDK

- Fix viewport resetting after each switch content on flat (non-headset) devices, even if the mesh type did not change.
- PC: increase incorrect initialize timeout (5 seconds) to the correct default value of 30 seconds.

v7.2.2-1-g6368f867 (2020-05-11)

Cross-platform

- Support self-signed certificates

Unity SDK

- PC: Fixed various crashes related to audio playback on Windows and Linux.

Native SDK iOS

- Fix CAMetalLayer not being resized when the ClearVRUIView is resized.
- The library can now be linked against iOS 9 and up, yet only iOS 11 and up are supported on run-time.

Native SDK Android

- Fix video being flipped upside-down when changing the device's orientation.

v7.2.1-5-g7b1149d4 (2020-04-22)

Cross-platform

- Fix playback of certain progressive MP4 clips being broken.

Unity SDK

- Fully compatible with Unity 2019 (verified on 2019.3.10f1)
- iOS: fix compatibility with Xcode 11.3.1, 11.4 and up
- PC: fix various crashes

Native SDK iOS

- Fix issue on Xcode 11.4.1 where `override convenience public init()` could not be resolved on `TMClearVRMotionOrientationTracker`

v7.2beta1-5-g1120dccf (2020-04-20)

Cross-platform

- Fix player getting stuck when seeking forward by a large amount
- Fix rectilinear video playback not always resumes after application suspend/resume cycle.
- Various DRM-related bugfixes
- Improved HLS playback stack

Unity SDK

- Beta support for Windows and Linux for NVidia graphics cards, including full editor support.

Native SDK iOS

- First public release of the new Native SDK for iOS.

v7.1.8-1-g4e4bcfd9 (2020-04-08)

Cross-platform

- Fix potential corrupt bitstream when hammering Seek()API on triple-GOP content
- Fix potential corrupt bitstream when hammering SwitchContent()API on triple-GOP content
- Minor optimizations to the HLS stack

Unity SDK

- Add SeekFlags.RelativeTime flag, allowing you to seek relative to the current position (e.g. +20 seconds). Using this flag allows internal optimization resulting in a much faster Seek() so please

make sure to use this when not doing an absolute seek

v7.1.5-2-ga9a4ec0f (2020-03-30)

UnitySDK

- Allow re-intializing ClearVRPlayer even if previous instance was not fully cleaned-up.
- Fix occasional hard crash when trying to play another clip after a player crash.
- Fix ClearVRPlayer not always properly getting destroyed in case of a FatalError.

v7.1.4-1-gb051bda7 (2020-04-01)

Cross-platform

- Improve HTTP/2 failed request handling
- Guard corner-case related to triple GOP content

v7.1.3-3-g320dcd91 (2020-03-30)

Unity SDK

- Fix occasional hard crash when trying to play another clip after a player crash.
- Fix ClearVRPlayer not always properly getting destroyed in case of a FatalError.
- Fix occasional temporary invalid HEVC bitstream output when calling SwitchContent() to triple GOP content.
- Android: passing an empty username as HTTP(S) proxy settings would incorrectly be interpreted as setting an invalid username.

v7.1.2-16-g0f98dd86 (2020-03-23)

Cross-platform

- Fix occasional crash in HLS playback stack

v7.1.2-13-gb78c416e (2020-03-20)

Cross-platform

- Support for content with longer GOP sizes, potentially improving visual quality and reducing bitrates.
- Support for seamless live event restarts. In the event of a problem at the production site, we can now restart the live feed without the need of making changes to the CMS.
- Fix a memory leak which could occur after using the SwitchContent() API many times in a row.

Unity SDK

- platformOptions.contentProtectionSystemhas been superseded by platformOptions.contentProtectionRobustnessLevel This will be relevant for playing back

Widevine L1/L3 protected content.

v7.1.1-5-g2cbe8141 (2020-03-20)

Unity SDK

- Android: add support for Widevine L3 protected content playout.
- Renamed ContentProtectionSystems enum to ContentProtectionRobustnessLevels enum

v7.1-203-geb331237 (2020-03-10)

Unity SDK

- Fix memory leak which could occur after many switchContent()
- Fix occasional A/V synchronisation issue with HLS playback

v7.1-113-g3cdfa6ee (2020-03-06)

Native SDK Android

- Based on our latest core libraries
- Add Support 180 degree, 360 degree and fish-eye ClearVR panorama video streaming
- Add fast content switching support
- Add magic window support
- Moved `com.tiledmedia.utils.Helpers` to `com.tiledmedia.clearvrhelpers`
- Various tweaks to the OpenGLSamplePlayer project:
- Compatible with API 29
    - Lock/unlock screen orientation toggle
    - Demonstrate how to toggle between supported Orientation Trackers
    - Demonstrate SwitchContent API usage
    - Code clean-up
- Drop support for the GearVRf framework
- clearVRPlayer.attachClearVROrientationTracker() API now throws an exception in case of failure.

Known issues:

- On some devices, upon enabling the magic window mode, you might experience an offset in the vertical plane.
- For unrelated legacy reasons, the libClearVRGearVRf library is still included in this package. You can safely ignore it, there is no need to add it to your project.

v7.0-gf122cc9 (2020-02-21)

Cross-platform

- Support for HLS and progressive MP4 content playback (HEVC+AAC only for now)
- Support for rectilinear ("16:9") content playback
- Support for traditional ERP180 (mono and SBS) and ERP360 (mono and TB) video playback
- Support for longer GOPs, potentially reducing bitrate and improving visual quality in case there is little headmovement
- Performance improvements reducing CPU load and memory pressure.

Unity SDK

- Android: support for Timewarp based rendering. This is experimental and requires OVR runtime v1.42.0
- A number of ClearVRPlayer APIs have been deprecated. These deprecated APIs have been decorated with an upgrade notice which should allow for easy migration
- The ContentItem has been upgraded with some additional fields. You can now specify a DRMInfo struct to pass the required DRM data (keys, passwords, license server, etc.). Furthermore, you can now override the projection type (contentItem.overrideContentFormat) when playing back non-ClearVR content (e.g. you have to tell the player whether your ContentItem is rectilinear or ERP180 SBS)
- Omnidirectional (180/360) video should be positioned at (0, 0 , 0) at all times like you have always done, rectilinear content can be freely placed using the new contentItem.startViewportAndObjectPose field.
- Switching between omnidirectional and rectilinear content is supported. Use the FirstFrameRendered event to keep track.
- There are two new shaders, and our three shaders in total have been moved from `Assets/ClearVR/Shaders/` to `Assets/ClearVR/Resources/Shaders/` As they are placed in `Resources`, you no longer need to explicitly through `Project Settings -> Graphics -> Always Included Shaders`.

MediaFlow Android

- Support for Widevine L1 protected content.

v6.0.3-1-gbbed16d8 (2019-12-06)

Android MediaFlow

- Fix poor playback performance on Samsung S10 devices that have installed the Android 10 system update.

v6.0.2-1046-gb7e8c443 (2019-11-29)

Cross-platform

- Fix VOD content restarting from 0 instead of resuming from the last know playback position after application lost/regained focus.
- Fix rare concurrency issue when trying to stop a player that is in the middle of Initializing.

Android: Fix rare corner case where broken content could result in a hard crash

v6.0.2-952-gd32a1922 (2019-11-25)

Cross-platform

- Live: fix issue where resuming playback after app focus is regained resulted in the player not resuming at the live edge. In some cases, the player could even get stuck in infinite loading.

v6.0.2-1-g679a7696 (2019-11-07)

Android MediaFlow

- Reduce adb logcat verbosity.
- Fix video decoder configuration issue on Pixel 3A.

v6.0.2-118-g3f97dc9a (2019-11-05)

Cross-platform

- Fix issue with SwitchContent where using the SeekFlag LiveEdge or WallclockTime did not always result in switching to the most current live edge.

v6.0.1-2-gc8870305 (2019-09-27)

Android MediaFlow

- Fix application occasionally freezing when switching content.

v6.0.1-2-gc8870305 (2019-09-25)

Unity SDK

- Fix application occasionally freezing when switching content.
- iOS: fix Metal rendering issues (shuffled tiles) sometimes being visible, including proper cardboard support.
- When using the switchContent() API, we now render a single mesh at all times. The mesh is no longer replaced upon switchContent() but rather adapted to fit the new content. This improves performance and stability.
- FirstFrameRendered is now triggered after every content switch and not only if the dimensions of the video changed.
- Various APIs w.r.t. mesh management have been deprecated and replaced by new ones. The deprecated APIs will remain around until 2020/01/31. It is quite likely that you were not using any of these APIs.
- Fix stereoscopic rendering not working on PicoVR-based headsets.

MediaFlow Android

- Improved decoder configuration mechanism, resulting in better support for devices that have buggy MediaCodec implementations
- Fix green-screen/partially green shuffled tiles issue on some lower end Kirin and -Snapdragon (6xx) chipsets.

v5.3.1-269-g53aff639 (2019-09-12)

Unity SDK

- Add official Metal support on iOS.
- Android: Fix green textures and broken/shuffled tiles on specific Kirin and Snapdragon 6xx devices.
- Mesh management has been overhauled. Previously, the mesh that rendered the video was created and destroyed upon every content switch. Now, there is only a single mesh throughout the lifetime of a ClearVRPlayer object which will morph with changing content.
- `ClearVREventTypes.FirstFrameRendered` is now triggered after every content switch and not only if the dimensions of the video changed.
- `clearVRPlayer.mediaInfo.GetDecoderVideoFrameWidth()` and `clearVRPlayer.mediaInfo.GetDecoderVideoFrameHeight()` have been replaced by `clearVRMesh.frameWidth` and `clearVRMesh.frameheight` respectively.
- `clearVRPlayer.mediaPlayer.SetMainColor()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.SetMainColor()` instead.
- `clearVRPlayer.mediaPlayer.Recenter()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.SetMainColor()` instead.
- `clearVRPlayer.mediaPlayer.Rotate()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.SetMainColor()` instead.
- `clearVRPlayer.mediaPlayer.Translate()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.SetMainColor()` instead.
- `clearVRPlayer.mediaPlayer.HideOrUnhide()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.HideOrUnhide()` instead.
- `clearVRPlayer.mediaPlayer.GetFallbackLayout()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.GetFallbackLayout()` instead.
- `clearVRPlayer.mediaPlayer.GetTexture()` has been deprecated and will be removed after 2020/01/31. Use `clearVRMesh.GetTexture()` instead.
- `clearVRPlayer.mediaPlayer.GetActiveClearVRMesh()` has been deprecated and will be removed after 2020/01/31. This API has become obsolete with the new mesh management mechanism.
- `PlatformOptions.camera` has been removed (deprecated since 2018/08/31). It has been replaced by `PlatformOptions.trackingTransform`
- `PlatformOptionsAndroid.isUseOESFastPathEnabled` has been deprecated. It has been replaced by `PlatformOptions.textureBlitMode`.
- Android: known limitation: PicoVR and WaveVR based headsets render stereoscopic video monoscopically only. The SDK will automatically disable stereoscopic playback on these devices to save on unnecessary data usage.

Android Native SDK

- No changes

iOS Native SDK

- No changes

Android MediaFlow

- Increased ClearVR bootstrap timeout that got inadvertently triggered when debugging your app through ADB.

iOS MediaFlow

- No changes

v5.3.1-25-gace32ad9 (2019-07-30)

Unity SDK

- iOS: implement audio control APIs: `SetMuteAudio()`, `GetAudioGain()`, `SetAudioGain()`, `GetIsAudioMuted()`
- iOS: implement `PreWarmCache()` API
- Android: fix stereoscopic rendering being broken on Oculus Quest.
- Android: add support for Wave VR SDK (stereoscopic playback will be fixed soon)
- Android: add support for Pico VR SDK (stereoscopic playback will be fixed soon)
- iOS: rename native plugin entry point to prevent function name collision with other third party libraries (was: UnityPluginLoad)

Android Native SDK

- implement audio control APIs: `setMuteAudio()`, `getAudioGain()`, `setAudioGain()`, `getIsAudioMuted()`

iOS Native SDK

- Add `PreWarmCache()` API

Android MediaFlow

- Improve inline documentation w.r.t. audio control APIs

iOS MediaFlow

- Add `PreWarmCache()` API
- Add audio controls
- Fix audio decoder crash on repeated switchContent

v5.2-4-g986c5d8f (2019-06-22)

Unity SDK

- Android: add beta support for WaveVR SDK based headsets like the HTC Vive Focus. Note that we only support WaveVR SDK v3.0.2 and up. Contact Tiledmedia for details.
- Android: improved support for the Oculus Quest.

Android Native SDK

- No changes

iOS Native SDK

- No changes

Android MediaFlow

- No changes

iOS MediaFlow

- Various stability improvements

v5.1.1-11-g8f4fd066 (2019-05-31)

Unity SDK

- Android: add support for arm64-v8a (ARM64) on Android. Note that all ClearVR android libraries (aar files) have been renamed and are now provided as "fat" binaries supporting both the arm-v7a and arm64-v8a architectures. After importing this new SDK, the previous version of these files will be automatically replaced.
- iOS: improved post-process build scripts for iOS builds on Unity 2018+
- Add PlatformOptions.http(s)ProxyParameters keys to hold proxy server information.
- Expose GetIsInPausedState() API to determine whether the player is in Paused state or not.
- Add GetDeviceAppId() API to retrieve the deviceAppId identifier.
- Fix issue where setting monoscopic/stereoscopic rendering interfered with what was actually available in the video bitstream.
- Fix potential lock-up when calling Stop() right after calling Initialize() (also known as the "player got stuck in stopping state" bug).
- Fix potential lock-up when calling Stop() right after calling SwitchContent().
- Fix potential lock-up when using SwitchContent() to rapidly switch between two clips only.

Known issues:

- iOS: visual glitches can be observed on Metal. We strongly recommend you to stick with OpenGLES for now.

Also refer to the Android MediaFlow and iOS MediaFlow section for all under-the-hood changes and known issues

Android Native SDK

- Redesigned the native SDK from the ground-up. It is now API compatible with the UnitySDK.
- Add preliminary support for the GearVRf framework.
- SDK is bumped to Android Studio 3.3 and targets Android API 28. Minimum support API level is still 23 (6.0 / Marshmallow).

Known issues:

- The GearVRf framework shows a verbose logline every vsync in logcat.
- The SwitchContent API is not yet functional and should not be used in this release.

Also refer to the Android MediaFlow section for all under-the-hood changes and known issues

iOS Native SDK

- Add support for setting proxy server settings.
- Add switch audio track functionality to sample player.
- Add seek functionality to sample player.
- Update documentation

Also refer to the iOS MediaFlow section for all under-the-hood changes and known issues

Android MediaFlow

- Fixed crash on Samsung S10 (all models) with latest firmware.
- Aligned APIs with MediaFlow-iOS
- Properly propagate proxy settings to underlying HTTP stack
- Regard "sw" (software) hevc decoders as unsupported because of their inadequate performance. This type of decoders is found on Snapdragon 801 based devices.
- Fix failing playback on certain Kirin-based devices.
- Improved video decoder configuration logic to work around various lower-end device-specific limitations.

Known issues:

- Android: the video can show up as being "shuffled" on some devices based on the Snapdragon 4xx and 6xx chipset.

iOS MediaFlow

- Properly propagate proxy settings to underlying HTTP stack.
- Fix crash in audio decoder.
- Fix bug where audio could start glitching after a Seek or Pause.

- Fix bug where calling Pause() while being in Paused state would throw a FatalError. This should've been a Warning only.
- Improved background-foreground handling.
- Improved chipset detection algorithm to properly detect decoder capabilities.

Known issues:

- Using the SwitchContent() API in rapid successions can result in a crash.

v4.6-33-g3f016f4a (2019-02-02)

- UnitySDK: Removed the following obsolete ClearVRPlayer APIs: `ClearVRPlayer.Pause()`, `ClearVRPlayer.Unpause()`, `ClearVRPlayer.StartPlayout()`, `ClearVRPlayer.ParseMediaInfo()`, `ClearVRPlayer.Stop(). Use the APIs on the` clearVRPlayer.controller` interface instead.
- UnitySDK: Removed the following obsolete ClearVREvent APIs: `ClearVREvent.ParseAudioTrackSwitchedEvent()` and `ClearVREvent.ParseClearVRCoreWrapperVideoDecoderCapabilitiesEvent()`. Use the equivalent APIs on `ClearVREvent.message` instead.
- UnitySDK: Removed the following obsolete ClearVRMessage APIs: `ClearVRMessage.ParseAudioTrackSwitched()`, use the renamed `ClearVREvent.message.ParseAudioTrackChanged()` instead.
- UnitySDK: Removed the following obsolete PlatformOptions fields: `camera`, use the `trackingTransform` field instead. `isTransparencySupportEnabled` has been removed as transparency is automatically detected when setting an alpha component != 0 using the `SetMainColor()` API.
- UnitySDK: Add support for planar and 180 degree videos
- UnitySDK: optimize cubic mesh to reduce polygon count by > 90%. This reduces GPU and CPU load, as well as improves texture sharpness.
- UnitySDK: Removed `TiledCubicSphereGameObject.cs` file, you can safely remove it from `Assets/ClearVR/Scripts/`.
- UnitySDK: Deprecated `ContentItem.hfovOffset`. Please do not use it anymore, its value is ignored in this release and the field will be removed completely from the SDK on July 31st, 2019.
- UnitySDK: Fix issues in ClearVRShader which resulted in no video showing up on the Samsung S9(+) on Android 9.
- UnitySDK: The UnitySDK is now generated using Unity 2018.3.0f2. Older versions of Unity might still work

v4.3-277-gbc48e2d4 (2018-12-14)

- MediaFlow: Add new `getArrayParameter()` and `getArrayParameterSafely()` APIs for `media.audio` parameters.
- UnitySDK: Add new `mediaPlayer.getClearVRCoreArrayParameter()` API for `media.audio` parameters.
- MediaFlow: Add getEventType() API to determine whether you are playing a VOD or LIVE clip.
- UnitySDK: Add `clearVRPlayer.mediaInfo.GetEventType()` API that can be used to determine

whether you are playing a VOD or LIVE clip. Check the `EventTypes` enum for this as well.

- UnitySDK: `clearVRPlayer.mediaPlayer.GetClearVRCoreVersion()` has been replaced with `static ClearVRPlayer.GetClearVRCoreVersion()`. This allows you to query the version of the core libraries at any point in time.
- UnitySDK: All shaders have been replaced with one shader that supports Single Pass and Multi Pass Stereoscopic Rendering (SPSR/MPSR), transparency (disabled if not used for optimal performance).
- UnitySDK: Add support for OES Fast Path texture in Single Pass Stereoscopic Rendering for improved performance.
- UnitySDK: Deprecated `platformOptions.isTransparencySupportEnabled` option. Transparency is now controlled by using the `SetMainColor()` API.
- UnitySDK: Breaking: remove `platformOptions.forceLegacyShaderForMultiPassRendering`. It was not used and is now superseded by the new shader.
- UnitySDK: `ClearVREventTypes.AudioTrackSwitched` and `ClearVREventTypes.StereoModeSwitched` no longer carry new audio track and stereo mode information respectively in their `String message` field in case of success. Use the new `ClearVRMessageCodes.ClearVRCoreWrapperAudioTrackChanged` and `ClearVRMessageCodes.ClearVRCoreWrapperStereoscopicModeChanged` `ClearVREventTypes.GenericMessage` `Info` messages to get the information that used to be relayed through previously mentioned `ClearVREvents`.
- MediaFlow: the entire external interface has been refactored. Most important changes:
  - `cbClearVRCoreWrapperGenericMessage(int, int, string)` renamed to `cbClearVRCoreWrapperMessage(ClearVRMessage)`
    - `cbClearVRCoreWrapperFatalError` has been removed. Messages are now relayed through the `cbClearVRCoreWrapperMessage(ClearVRMessage)` callback.
    - All request APIs like pause/unpause/seek/loadContent now have an asynchronous implementation with inline callback interface and support for optional arguments. This allows you to receive a callback to each requests.
    - New `cbClearVRCoreWrapperRequestCompleted(ClearVRAsyncRequestResponse)` that is called if a request received a response but no inline callback interface was specified.
    - Messages previously consisted of a `messageType`, `messageCode` and `message` fields. These are now grouped into a convenience `ClearVRMessage` object.
    - `messageCode` is still an `int`, yet it will hold either a value from `enum ClearVRMessageCodes` or the new `enum ClearVRCoreErrorCodes`. Use the exposed APIs to figure out what value the code represents.
    - `messageType` has been changed from an `int` into an `enum ClearVRMessageTypes` field.
    - `enum ClearVREventTypes` and `enum ClearVRStereoscopicModes` have been added to enumerate event type (e.g. VOD or LIVE) and stereoscopic mode (e.g. monoscopic or stereoscopic) respectively.
    - `void clearVRCoreWrapper.start()` has been renamed to `ClearVRAsyncRequest initialize(callback, ...)`.

    - `void clearVRCoreWrapper.startClearVRCore()` has been renamed to `void clearVRCoreWrapper.loadContent(callback, ...)` to better reflect what it is actually doing.

- UnitySDK: The UnitySDK now has an asynchronous API with inline `Action` based callbacks, in line with the changes to the MediaFlow library (see above).
- UnitySDK: Add `Demo02` sample scene that will supersede Demo01 scene and script.
- OpenGLSamplePlayer: refactored code to be in line with the changes mentioned above. Implemented callbacks to better separate video player rendering from video player logic and control-flow. See the source code for details.

v4.2-1-gb9ee2d88 (2018-10-23)

- UnitySDK: Moved `mediaPlayer.PrepareCore()` to `internal.PrepareCore()` as it is a private API that should not have been exposed in the first place.
- MediaFlow: Renamed `interFrameApplicationLatencyTrackerMean` to `interFrameApplicationLatencyMean` for consistency. Same holds for its StandardDeviation counterpart.
- UnitySDK: Renamed `ClearVRPlayer.PausePlaybackAfterAppFocusOrPauseChangedLegacy()` to `ClearVRPlayer.PauseOrUnpausePlaybackAfterAppFocusOrPauseChangedLegacy()` to better fit the methods implementation.
- MediaFlow: the following methods have been removed:
  `void startClearVRCore(ContentItem argContentItem, int argAudioDecoderType, int argAudioPlaybackEngineType)`
  `void startClearVRCore(final ContentItem argContentItem)`
  `void startClearVRCore(String argManifestUrl, float argStartYaw, float argStartPitch, long argStartPositionInMilliseconds)`
- MediaFlow: the following method has been added:
  `void startClearVRCore(StartClearVRCoreParameters argStartClearVRCoreTimeoutInMilliseconds)`. Furthermore, `StartClearVRCoreParameters` is now exposed as a public class.
- MediaFlow: one can now override the timeout triggered when loading content takes too much time through the `StartClearVRCoreParameters.startClearVRCoreTimeoutInMilliseconds` field.
- UnitySDK: Add `mediaPlayer.Translate()` API to translate the transform of the GameObject.
- UnitySDK: Add support for transparency by setting `platformOptions.isTransparencySupportEnabled = true`. By default, this is disabled due to the potential performance impact a transparent shader might have.
- UnitySDK: The UnitySDK has been developed in Unity 2017.4.10f1. Backwards compatibility goes back to at least Unity 2017.3.0f1.
- UnitySDK: Fix problem where a `ClearVRPlayer` would always reload the clip it was initialized with after the application lost focus (with `ApplicationFocusAndPauseHandlingTypes.Recommended`), even if another clip was playing through the `SwitchContent()` API.
- UnitySDK: add support for transparency by setting `platformOptions.isTransparencySupportEnabled = true`. By default, this is disabled due to the

potential performance impact a transparent shader might have.

- UnitySDK: the UnitySDK has been developed in Unity 2017.4.10f1. Backwards compatibility goes back to at least Unity 2017.3.0f1.
- UnitySDK: fix problem where a `ClearVRPlayer` would always reload the clip it was initialized with after the application lost focus (with `ApplicationFocusAndPauseHandlingTypes.Recommended`), even if another clip was playing through the `SwitchContent()` API.
- UnitySDK: `platformOptions.camera` is deprecated in favor of `platformOptions.trackingTransform`. Please update your code accordingly.
- UnitySDK: Add `mediaPlayer.GetIsInStoppedState()` API to check whether the `clearVRPlayer` is in `Stopped` state or not.
- UnitySDK: Expose `ClearVRPlayer.DestroyMediaPlayer()` API which allows one to destroy the `mediaPlayer` currently active on the `ClearVRPlayer` object. Note that one can ONLY call this method when the ClearVRPlayer is in `Stopped` state. It is up to the implementer to make sure that it is in said state.

v4.0beta1-806-ga4ffdd8 (2018-09-18)

- UnitySDK: Fix bug where rendering would briefly switch to monoscopic first if switching from a monoscopic to stereoscopic clip.
- UnitySDK: Fix bug where the right eye would be empty for one frame is switching from a monoscopic to stereoscopic clip.
- UnitySDK: Fix a rare concurrency segfault when destroying a ClearVRPlayer while at the same time new UVs were being calculated in the NativeRendererPlugin.
- UnitySDK: Fix initial camera offset not being taken into account when starting a ClearVRPlayer and when using the `SwitchContent()` API. This resulted in occasional needless buffering and cache trashing just after initialize and/or switch content completed.
- MediaFlow: expose new `cachePrewarm()` API allowing one to preload metadata of a content item. This includes the new `cbClearVRCoreWrapperPrewarmCacheCompleted()` callback.
- UnitySDK: expose new `CachePrewarm()` API, see previous item on this list.
- UnitySDK: `GetDecoderVideoFrameWidth()` and `GetDecoderVideoFrameHeight()` moved from the `performance` interface to the `mediaInfo` interface.
- MediaFlow: Improved playout smoothness of p30 content by explicitly locking to the vsync.
- MediaFlow: Add support for p60 content playback.
- MediaFlow: Removed APIs: `getEncodedAndDecodedVideoQueueSizesAsString()`, `getDecoderLatencyMeanAndStddev()`, `getAverageFrameRenderTime()`, `getVideoFramerate()`, `getStreamByStreamType()`. Please use the new `getClearVRCoreWrapperStatistics()` API to get access to these values.
- UnitySDK: Removed APIs: `GetAverageFrameRenderTime()`, `GetEncodedAndDecodedVideoQueueSizes()` and `GetDecoderVideoFramerate()`. Please use the new stable `ClearVRPlayer.performance.clearVRCoreWrapperStatistics` field to access performance statistics. Note that this field is `null` while the player is initializing.

v4.0beta1-1-eca6816 (2018-07-23)

- UnitySDK: `switchContent()` API now supports switching to content that has a different resolution and/or framerate. For example, you can now switch between monoscopic 8K and stereoscopic 6K playback on the fly without the need to destroy and create a new ClearVRPlayer object.
- UnitySDK: Breaking change: renamed `.Play()` to `.StartPlayout()` to avoid confusion on when to call this method. Note that you can only call `.StartPlayout()` once on a `ClearVRPlayer` object.
- UnitySDK: Added TM_MESSAGE_CLEAR_VR_CORE_INITIALIZATION_TIMEOUT message code, triggered when a timeout occurs while loading new content.
- ClearVRMediaFlow: increased initialization timeout from 15 seconds to 90 seconds to accommodate slow networks.
- UnitySDK: The `TiledCubicSphere` mesh is now completely generated in C++, whereas previously it was both partly generated in C# and partly in C++. From the point of the integrator nothing has changed but this has reduced code duplication and performance improvement when constructing the `TiledCubicSphere` mesh during initialization.
- OpenGLSamplePlayer: the OpenGLSamplePlayer has been ported and no longer depends on protobuf. Inline documentation has been updated and the player has been updated to be compatible with the latest APIs as exposed by `ClearVRCore` and `ClearVRMediaFlow`.
- ClearVRMediaFlow: Breaking API change: `startClearVRCore()` and `switchContent()` no longer accept hfov and vfov as parameters. The Core library will determine "good" values for hfov and vfov itself. The `ContentItem()` class also no longer accepts these parameters. If one wants to override ClearVRCore's default hfov/vfov values, one should set them *before* `initialize()` is called by setting 'config.hfov_decode_buffer' and 'config.vfov_decode_buffer' respectively. These values are interpreted as an offset to the default values, e.g. +20 will increase the default respective fov with 20 degrees. Note that the hfov and vfov used during initialization are the maximum value as well. You can never set them to a higher value later on.
- UnitySDK: Add `ClearVREventTypes.ContentFormatChanged` event which fires as soon as the ContentFormat has changed (e.g. one switched from a monoscopic to stereoscopic clip with the `switchContent()` API). Note that this is something else than `RenderMode`.
- UnitySDK: Removed `MediaInfoInterface.GetIsContentStereoscopic()` as it has been replaced with `MediaInfoInterface.GetContentFormat()`.
- UnitySDK: `ClearVREventTypes.FirstFrameRendered` is now emitted after every first frame rendered on a *new* sphere (e.g. after a content switch to content with a different resolution).

v3.2-3-gf957956 (2018-06-28)

- UnitySDK: Renamed `ApplicationFocusAndPauseHandlingTypes.LEGACY`, `.RECOMMENDED` and `.DISABLED` to camel-case (`Legacy`, `Recommended`, `Disabled`) for consistency reasons.
- UnitySDK: Add support for Single Pass Stereo Rendering (SPSR) for improved performance. Actual performance improvement may vary depending on scene complexity. Note that SPSR is in preview in Unity 2017.x so please check your application extensively. Your mileage may vary from device to device.
- UnitySDK: Add `ClearVREventTypes.UnableToInitializePlayer` event, triggered when something goes wrong while initializing the player.

- UnitySDK: Fixed issue with `ApplicationFocusAndPauseHandlingTypes.Recommended` not properly resuming playback after application was paused.
- UnitySDK: Fixed `SwitchContent()` not taking start position into account. Playback would always start at 0.
- UnitySDK: Various Oculus Go related fixes.
- ClearVRMediaFlow: Added quirk for a specific Broadcom video decoder which always holds on to the last decoded frame, causing the decoder priming to timeout and playback to fail.
- ClearVRMediaFlow: Prevent decoder output queue from overflowing when the application as well as the core is paused (which would happen if the application would be pushed to the background). Note that after resuming the application, a hand-full of dropped frames warnings are signaled to the application (see also the next bullet). This only applies when you have set `ApplicationFocusAndPauseHandlingTypes.Legacy`.
- ClearVRMediaFlow: Added TM_MESSAGE_VIDEO_DECODER_FRAME_DROPPED (-1108) message code to signal a dropped frame. Previously, this message had a TM_MESSAGE_UNSPECIFIED_WARNING message code.
- UnitySDK: `ClearVRMessageCodes.ClearVRCoreWrapperVideoDecoderFrameDropped` warning message code to signal dropped video frames (see previous bullet).
- UnitySDK: Allow to override the parent gameobject to which the `TiledCubicSphere` gameobject should be bound by setting `platformOptions.parentGameObject` accordingly.
- ClearVRMediaFlow: Add support for `SetStereoMode()` API to dynamically switch between mono and stereo at runtime. The new `ClearVREvents.StereoModeSwitched` will notify you if the switch was successful and whether rendering switched to monoscopic or stereoscopic.
- ClearVRMediaFlow: API change: `clearVRCoreWrapper.startPlayoutAfterContentLoadingCompleted()` now returns `true` in case of success or `false` otherwise.
- ClearVRMediaFlow: Add `getParameterSafely()` and `setParameterSafely()` methods. The non-Safely variants have a `throws Exception` signature which is incompatible with Mono/C# (the exception cannot be propagated from Java to C#). For now, the developer should be aware of any error written to logcat.
- ClearVRUtils: This new library exposes a number of low-level APIs w.r.t. thread management.
- UnitySDK: we now force ClearVRCore (or more precise: the Go runtime) to only run on high performance cores if available. This will alleviate low power cores and fixes an issue where rendering thread starvation could occur if `ClearVRCore` would run on the same core as the rendering thread. This would result in severe tearing and stale frames on a GearVR or Oculus Go.

v3.2rc3 (2018-05-25)

- UnitySDK: Breaking change: dropped support for the Oculus Media Surface Plugin. The ClearVR Native Renderer Plugin, which has been the default plugin for the since the end of 2017, has more features and has significantly better performance.
- UnitySDK: add ability to override default sphere radius.
- UnitySDK: add ability to override default sphere `GameObject` layer.

- Breaking change: the SDK will refuse playout if no hardware HEVC decoder is detected.
- Significantly reduced memory usage resulting in less time spent in garbage collection and improved performance.
- Many improvements to reduce the chances of shuffled tiles.
- UnitySDK/ClearVRSDK: No longer require protobuf. Protobuf has been dropped as a dependency and can be removed from your project. Furthermore, `TMInterProcess.cs` can be removed from your Unity project as well (located in Assets/ClearVR/Scripts/)
- UnitySDK: split TiledCubicSphere script in two scripts: one holding Unity-specific logic and one holding mesh specific logic.
- UnitySDK: Android: switched to asynchronous UV calculations, alleviating the main Unity thread to reduce frame time jitter.
- UnitySDK: The TiledCubicSphere radius can now be specified, as well as the layer it will be rendered to. Use the `PlatformOptions` struct for this
- UnitySDK: The TiledCubicSphere is now directly attached to the game object created by the ClearVRPlayer script. An extra game object is no longer created.
- ClearVRMediaFlow: `TM_VIDEO_DECODER_CANNOT_DECODE_FRAME` (-1013) was renamed to `TM_MESSAGE_VIDEO_DECODER_CANNOT_DECODE_FRAME` for consistency reasons.
- ClearVRMediaFlow: Added `TM_MESSAGE_VIDEO_DECODER_DOES_NOT_SUPPORT_PROFILE_OR_LEVEL` (-1014) error, indicating that the available hardware decoder does not meet the requirements to decode the selected video clip. Note that this error is fatal, the library will refuse playback if this event is triggered.
- UnitySDK: revised application focus and pause handling. Video playback will now be stopped when the application is paused and pushed to the background (e.g. triggered by Unity's OnApplicationPause()) to conserve battery power. The message field of the `ClearVREventTypes.StateChangedStopped` event now holds the content time in milliseconds at the moment payback was stopped. You can use this time to resume playback after the application regained focus. See the demo scene on how to access this information. Legacy behavior (just pausing playback) is still available but not recommended as it will drain your battery. The new `platformOptions.applicationFocusAndPauseHandling` and `ApplicationFocusAndPauseHandlingTypes` enum can be used to select the mode of operation. You can completely disable application focus/paused handling by setting this value to `ApplicationFocusAndPauseHandlingTypes.DISABLED`. This is not recommended though!
- UnitySDK: Moved and renamed `Utils.SplitAudioTrackSwitchedMessage()` to `Types.ClearVREvent.ParseAudioTrackSwitchedEvent()` for consistency reasons.
- UnitySDK: Added informational `ClearVRMessageCodes.ClearVRCoreWrapperVideoDecoderCapabilities` (-1202) which includes information on the video decoder in use. Use `Types.ClearVREvent.ParseClearVRCoreWrapperVideoDecoderCapabilitiesEvent()` to parse the message.

Known issues:

- On very bad networks and after many successive buffering -> running cycles, a lag of one additional video frame might be noticed from time to time. We are aware of this problem and will be fixed in the next release.
- the OpenGLSamplePlayer project is missing from the SDK. It will be re-added in the next release.

v3.1rc1-1-g226039d (2018-04-23)

- Stereoscopic rendering has been fixed in this release.
- Fixed seek-crash.
- Android: fixed non-fast OES mode crashing on some devices.
- UnitySDK: Add static `ClearVRPlayer.GetIsPlatformSupported()` and `ClearVRPlayer.GetIsPlatformSupported(UnityEngine.RuntimePlatform argPlatform)` methods to query whether ClearVR supports the current platform / a specific platform respectively.
- Fix slight audio/video desynchronisation issue when playing back specific content.
- Applied various fixes for shuffled tiles on slower devices.
- UnitySDK: Add `clearVRPlayer.performance.GetDecoderVideoFrameWidth()` and `clearVRPlayer.performance.GetDecoderVideoFrameHeight()` methods to query the *decoder* video frame width and height in pixels.
- ClearVRMediaFlow: Add `clearVRCoreWrapper.getNumberOfAudioTracks()` helper method to query the number of available audio tracks. Returns -1 is unknown.
- ClearVRMediaFlow: `clearVRCoreWrapper.setAudioTrack()` now has a sibling method signature allowing you to override the `AudioDecoderType` and `AudioPlaybackEngineType` when switching to a different audio track.
- UnitySDK: `clearVRPlayer.controller.setAudioTrack()` now has a sibling method signature allowing you to override the `AudioDecoderType` and `AudioPlaybackEngineType` when switching to a different audio track.
- ClearVRMediaFlow: added `cbClearVRCoreWrapperAudioTrackSwitched()` callback to `ClearVRCoreWrapperExternalInterface` which is triggered when the audio track is switched. It's callback message will hold three comma-separated integers, e.g. 1,0,1 representing audio track index, audio decoder type and audio playback engine type respectively. All three numbers will read -1 when no audio track is selected.
- UnitySDK: Expose `setAudioTrack` API. Added `cbClearVRCoreWrapperAudioTrackSwitched()` callback and `ClearVREventTypes.AudioTrackSwitched` event.
- UnitySDK: Add `GetNumberOfAudioTracks()` helper method on `mediaInfo` interface.
- ClearVRMediaFlow: add `switchContent()` API to quickly switch between different content items. This is currently limited to content with equal resoution and framerate.
- UnitySDK: Expose `switchContent` API. Added `cbClearVRCoreWrapperContentSwitched()` callback and `ClearVREventTypes.ContentSwitched` event.
- UnitySDK: Add `clearVRPlayer.mediaPlayer.SetMainColor()` API to change the main color of the `TiledCubicSphere` material
- UnitySDK: Add `ContentItem.startYaw` and `ContentItem.startPitch` members to manually specify

head orientation at the moment that content is about to be loaded. Note that the programmer has to set these values himself, both members are defaulted to 0 (degree).

v3.0-1002-g1c57929 (2018-03-31)

- Added ClearVR.unitypackage file containing all required C# scripts, shaders, libraries and AndroidManifest.xml. This will greatly simplify deployment in your own project and upgrading your project when a new version of our SDK is released.
- Added `Stopped` state (which replaces `Finalized` state). Renamed `Finalized` to `Stopped` and added `Finished` as extra state. `Finished` state is entered when the end of the content has been reached.
- `media.effective_content_resolution` key has been deprecated and superseded by `media.highest_quality`. Accordingly, `clearVRPlayer.mediaInfo.GetEffectiveContentResolution()` has been renamed to `clearVRPlayer.mediaInfo.GetHighestQualityResolutionAndFramerate()` which returns dimension and framerate of the highest quality representation layer available.
- It is now possible to set the (approximate) start position of the content by specifying `contentItem.startPositionInMilliseconds` accordingly.
- Add `ClearVREventTypes.FirstFrameRendered` informational ClearVREvent that is triggered as soon as the very first video frame is actually rendered on the sphere.
- UnitySDK: Add `clearVRPlayer.DebugInterface` which will contain debug methods that can change from time to time. You are highly encouraged to NOT use the exposed methods in production since they can change from release to release without any notice.
- UnitySDK: Add `clearVRPlayer.controller.TogglePause()` method that will toggle between pause and unpause depending on the current state.

Known issues:

- You will see one warning line reading `E/Unity : OPENGL NATIVE PLUG-IN ERROR: GL_INVALID_ENUM: enum argument out of range` warning when the content is loading. This warning is harmless and can be ignored for now but will be fixed in a future release.
- Stereoscopic rendering will glitch in certain head orientations.

v3.0-617-g3c80741 (2018-03-14)

- Refactored Unity3D SDK. ClearVRForUnity has been stripped down to just simply making sure that the required libraries are loaded as soon as possible. No ClearVRCoreWrapper method passthrough is happening anymore in this library.
- Use `AndroidJavaProxy` to facilitate callbacks from the Android library into Unity3D instead of relying on `GameObject.SendMessage()`.
- Restructured folder layout to make inclusion in your own project more straightforward. See Assets/ClearVR/ for all ClearVR related scripts, libraries and a demo scene + script.
- One simple controller interface without platform (iOS/Android/PC) specific dependencies opening the way for easy cross-platform integration.

- Split sample implementation from SDK-wrapper interface making life a lot easier for you as an integrator.
- Subscribe to an event channel to get notified of all state changes (playing/pausing/buffering/etc.) and other noteworthy events.
- Single `TiledCubicSphere` for both mono and stereo content.
- Shader based toggling between mono and stereo.
- No more clunky layers and elaborate monoscopic -> stereoscopic switching mechanism using the `OVRManager`.
- Monoscopic and stereoscopic rendering performance improvements resulting in a smoother playback and less dropped frames.
- Many ClearVRCore related performance improvements.
- Added seeking while in paused.
- Fixed occasional A/V sync issues.
- OpenGLES 2.0 and OpenGLES 3.0 support (single-pass rendering is not supported)
- Daydream support (at least on Nexus devices).
- Dropped MediaSurfacePlugin in favor of our own C++ based implementation (although it is still supported as a fallback option in this release).

v3.0-5519b34 (2018-01-12)

- UnitySamplePlayer: Split initialization phase of the first video frame over multiple rendered frames to reduce frame judder when loading a content item.
- UnitySamplePlayer: Renamed `getAverageBitrateInMbps()` to `GetAverageBitrateInMbps()` (capitalization consistency)
- UnitySamplePlayer: Renamed `GetLibClearVRCoreParameter()` to `GetClearVRCoreParameter()` (naming consistency)
- UnitySamplePlayer: Renamed `SetLibClearVRCoreParameter()` to `SetClearVRCoreParameter()` (naming consistency)
- UnitySamplePlayer: demonstrate pause/unpause when application is pushed to the background/looses focus.
- Add OpenGLSamplePlayer JAVA-based video player source code to SDK.
- Revised ClearVRSDK documentation to include the OpenGLSamplePlayer.

v3.0rc1-2d90461 (2017-12-27)

- Breaking change: libRecombiner has been renamed to ClearVRCore (or libClearVRCore where applicable). Due to this change, a number of APIs has changed.
- Breaking change: libTmMfc has been renamed to ClearVRMediaFlow.
- Breaking change: libTmUnity has been renamed to ClearVRForUnity. The package name has been updated from `com.tiledmedia.libtmunityandroid` to `com.tiledmedia.clearvrforunityandroid` and the main activity (extending `UnityPlayerActivity`) has been renamed from `com.tiledmedia.libtmunityandroid.TmUnityAndroidActivity` to `com.tiledmedia.clearvrforunityandroid.ClearVRForUnityAndroidActivity`. Make sure you update your

`AndroidManifest.xml` file accordingly!

- Breaking change: fix issue where EXTERNAL_TEXTURE_OES was not properly released by the OculusMediaSurface plugin. This resulted in a black texture when more than one video was played back at the same time (e.g. when using the third party MovieTexture plugin alongside our library).
- Breaking change: CubicSphereShader shader has been updated as part of the previous change.
- Breaking change: add support for stereoscopic rendering. Note that the HUD text may be difficult to focus on in some occasions. It is provided for demonstrative purposes only.
- SDK now comes with two precompiled APKs of the UnitySamplePlayer (both flat and GearVR). These precompiled APKs expect your license file to be present in the root of the phone's storage (not on the SD card).
- The UnitySamplePlayer project now includes all required dependencies, including the latest Oculus Utilities Unity3D package.
- Fixed bug where the library would not pass initialization when ran on an NVidia Tegra based system (found on e.g. the Nvidia Shield).
- Fixed button alignment issue in UnitySamplePlayer.
- Implemented `ClearVRCore.CrashHandler()` to catch panics in our core library. The panic will be converted to a Java `RuntimeException()` making sure that any application level crash-reporting tool (like Crashlytics for example) can catch it. This is a workaround for the fact that Golang code does not allow proper panic trapping right now on Android and iOs.
- Expose head-orientation (yaw/pitch/raw) in the AudioDecoderThread class to ease the integration with spatial audio engines.
- Include flow chart in the documentation detailing the general flow order of execution.

v2.12-129-ga28cf5d (2017-11-29)

- Implemented internal AudioDecoderThreadInternalInterface interface to properly report audio decoder status to LibRecombinerWrapper.
- Added TM_MESSAGE_AUDIO_DECODER_DECODING_FAILURE (-1021) error message when an error is reported by the audio decoder.
- Fixed index out of bounds error which has been reported to occur on Samsung S7 (Edge) phones running Android Marshmallow.
- The Unity Sample Player code has been cleaned-up. Demonstrates how to stop and start content.
- Head orientation is now exposed to the AudioDecoderThread class allowing for facilitate the future integration of spatial audio.
- Added `AudioTrackProperties` class to allow for simple communication of audio track properties (e.g. hoa settings like channel ordering and format). This can benefit the implementation of spatial audio. As such, the `AudioDecoderThread` arguments have slightly changed.
- `LibRecombinerWrapper` no longer automatically selects the first non-higher-order-audio track. This is up to the application from now on. Selecting the correct track can be accomplished using

`parseMediaInfoAsync()` and get-ing and set-ing a couple of parameters (e.g. `config.audio_track`, see libRecombiner documentation). See also the next bullet.

- The `LibRecombinerWrapper` class no longer defaults to `config.audio.playout_offset = -200`. This should be handled by the application from now on. Due to audio decoder and playout latency (which typically hovers around 180-200msec) audio is typically handled **earlier** than video. You should use the `parseMediaInfoAsync()` method to query the media information and, subsequently, select the audio track you need. `parseMediaInfoAsync()` can be called after `LibRecombinerWrapper` class has been initialized (and a `libRecombiner` context has been created). `parseMediaInfoAsync()` is completed asynchronous since it can take a couple of 100 msecs to complete. Wait for the TM_MESSAGE_LIB_RECOMBINER_MEDIA_INFO_PARSED (-10010) callback in the application to know when it has completed. See the updated Unity Sample Player for an example and further details.

v2.12-102-g2939ef4 (2017-10-27)

- MediaCodecCapabilities class added which determines whether an hardware HEVC decoder is present or not and its capabilities.
- Expose error message to application layer if HEVC decoder throws an error while decoding a frame.
- libRecombinerWrapper.setParameter() now throws an error when no libRecombiner context is created.
- libRecombinerWrapper.getParameter() now throws an error when no libRecombiner context is created instead of returning an empty string.
- ClearVRForUnity.initialize() no longer takes argApplicationContext argument. Note that the LibRecombinerWrapper() class still requires an argApplicationContext argument in case you are using low-level integration.
- Added `ClearVRForUnity.getIsHardwareHevcDecoderAvailable()` and `ClearVRForUnity.getIsSoftwareHevcDecoderAvailable()` methods to determine HEVC decoder availability.
- Added `ClearVRForUnity.getEffectiveContentResolution()` method to retrieve the actual (source) resolution.
- Renamed `LibRecombinerWrapper.getIsAtLeastInitialized()` to `LibRecombinerWrapper.getIsAtLeastInitializedAndNotFinished()` because of added `Librecombiner.RecombinerStateFinished` state.
- LibRecombinerWrapper() class constructor takes a new argDeviceAppId string argument. Leave this argument blank to let the underlying library generate a unique device and app specific id for you or provide one yourself.
- Include MediaSurfacePlugin source code in Android SDK.

v2.11-118-gbbcf332 (2017-09-06)

- Based on libRecombiner v2.11 with numerous performance improvements.
- Added Unity specific Interface (LibRecombinerWrapperExternalInterfaceForUnity) which

extends LibRecombinerWrapperExternalInterface and includes a standardized initialize() method.
- Bug fix: player sometimes became unresponsive due to improperly closed audio and video decoder thread not freeing resources when waiting for next frame which would never come (because we were in the process of shutting down). This only happened when the a LibRecombinerWrapper instance was reused between runs.
- Add a bit of delay after the video decoder has been primed before actually signaling the application that this step has completed. This will allow the application to finish setting things up (e.g. constructing its sphere).
- Implement drop-based frame scheduling algorithm in TimeModel rather than a wait-for-frame-based frame scheduling algorithm for video. This will yield smoother playback at the cost of an occasionally dropped frame.
- In prior release, initialization of the underlying libraries meant that the content would also start playing immediately after load completed. You will now receive callbacks between each step of the process on the application level, allowing you to intervene where need be.
- TM_MESSAGE_LIB_RECOMBINER_INITIALIZED (-10000), TM_MESSAGE_LIB_RECOMBINER_READY_FOR_PLAYOUT (-10001) and TM_MESSAGE_LIB_RECOMBINER_UNINITIALIZED (-10002) messages now have a meaningful message field (which was previously just empty). Check for "OK" if all is ok; this message will contain an error message in case a problem occurred.

v2.10.1-71-g110226b (2017-08-14)

- First release of SDK
- Based on libRecombiner v2.10.

Upgrading to v10

This section describes the upgrade cycle from v9.x to v10.x. If your project is still using an older SDK, please refer to the Upgrading to v9.x section first.

CbClearVREvent handling: call RemoveListener

One subscribes to the events emitted by the ClearVRPlayer object, e.g.:

```
clearVRPlayer.clearVREvents.AddListener(CbClearVREvent);
clearVRPlayer.clearVRDisplayObjectEvents.AddListener(CbClearVRDisplayObjectEvent);
```

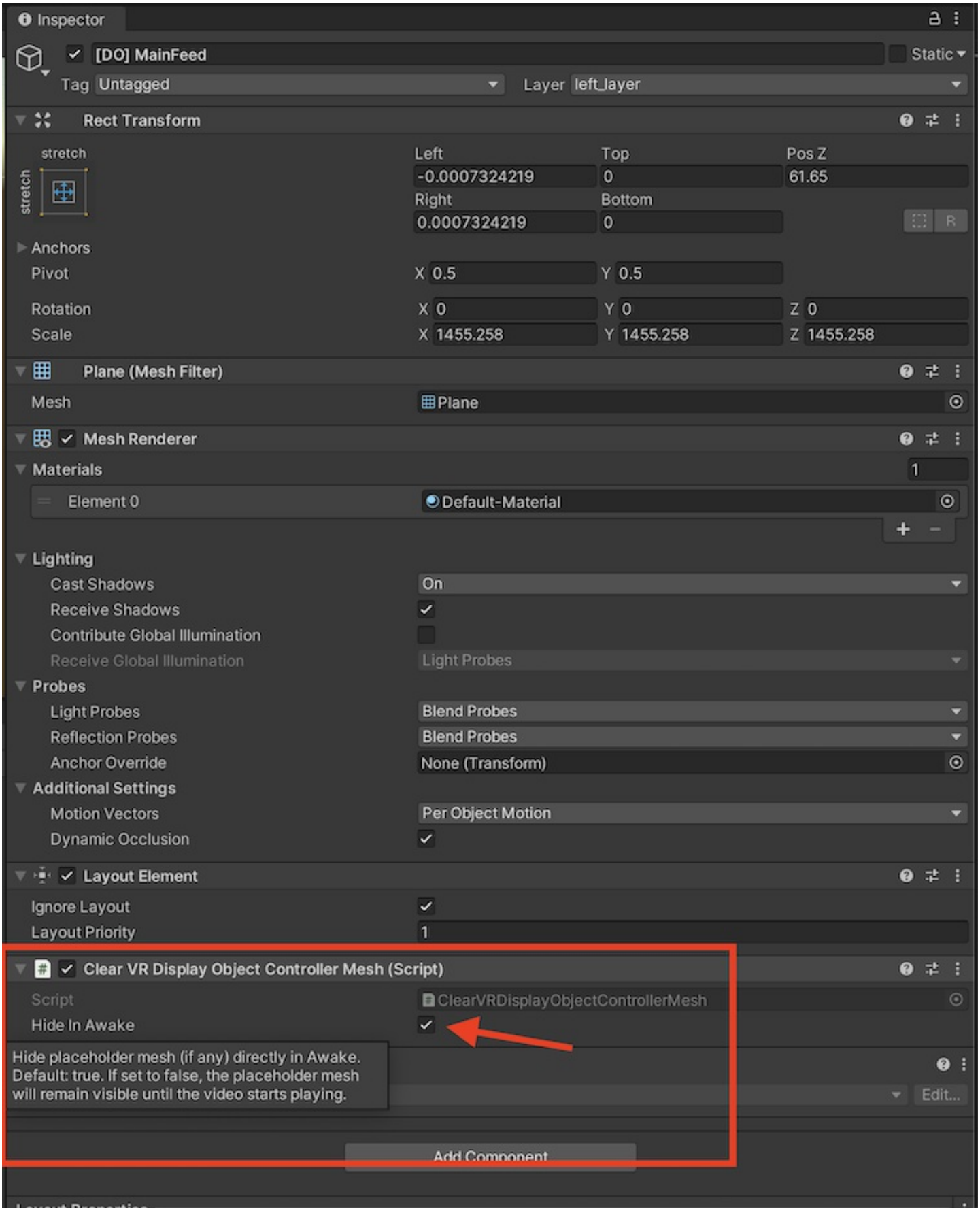
Developers that based their implementation on the UnitySamplePlayer project that comes along with the ClearVR/Tiledmedia Player SDK should be advised that some versions of this sample player project lacked unregistering the listeners. As a result, it could happen that your application would receive the same event multiple times *if you were reusing* the same `ClearVRPlayer` component after a Start/Stop cycle.

You can update your code to call `RemoveListener` in `StateChangedStopped`, e.g.:

```
switch (argClearVREvent.type) {
    ...
    case ClearVREventTypes.StateChangedStopped:
        ...
        clearVRPlayer.clearVREvents.RemoveListener(CbClearVREvent);
        clearVRPlayer.clearVRDisplayObjectEvents.RemoveListener(CbClearVRDisplayObjectEvent);
        break;
    ...
}
```

Display object managment

In v9.x, While loading content, display objects with a placeholder mesh would be visible for a very short instant of time (one or two frames). In v10, all display objects will fully disappear from the very first frame by default. They will show up once the feed for that display object is loaded. This default behaviour can be overridden per display object, see the screenshot:



Subtitle support added -> update your LayoutParameters

v10.0 adds subtitle support. Subtitle tracks are treated similar to audio tracks. If you are migrating

from v9.x, you should explicitly set the subtitle you want to render. To trigger default behaviour, add the following line:

```
layoutParameters.subtitleTrackID = null;
```

and, in the LayoutManager inspector panel, set `subtitle feed index` and `subtitle track index` to -2 for auto-select.

iOS

Apple has deprecated Bitcode support in XCode 14. As a result, our libraries are no longer built with it.

Gathering verbose Tiledmedia SDK logs

Sometimes, we ask you to temporarily enable verbose logging when you run into an unforeseen problem. In v10.x, we have overhauled the logging mechanism and made it easier for our customers to configure the logging for debugging purposes. You used to enable verbose logging using (a derivative of) the following code:

```
ClearVRPlayer.coreLogLevel = 2;
ClearVRPlayer.coreLogFile = "/storage/emulated/0/clearvrcore.log";
```

In v10.x, to enable verbose logging, the above code has been replaced by:

```
LoggingConfiguration loggingConfig = LoggingConfiguration.GetDefaultLoggingConfiguration()
ClearVRPlayer.EnableLogging(loggingConfig);
```

This will write verbose logging (`.tmlog`) and event recorder files (`.tmerp`) to disk. To locate the output folder, check the console/logcat. In Unity, the default folder equals `Application.persistentDataPath`.

> **⚠ WARNING**
>
> As in previous versions, enabling verbose logging incurs a (significant) performance penalty. Be sure to *never* enable this logging by default in your production application.

Uploading Tiledmedia SDK logs

To facilitate customer-side debugging of potential application and/or Tiledmedia SDK issues, besides enabling verbose logging, v10.0 adds the option to upload gathered log files to the Tiledmedia backend. To use this feature, one would:

1. Enable verbose logging (see the previous section).

2. Trigger the problem.
3. Call the UploadLogs API, you can do this e.g.:
    - as soon as the problem has happened (e.g. in a callback or after a specific `ClearVREvent`).
    - at the end of the run (e.g. after you have reached `StateChangedStopped`).
    - at the beginning of the next run (for example if the app or the Unity Editor crashed because of the problem you ran into).

Example code:

```
ClearVRPlayer.UploadLogs(YOUR_LICENSE_FILE_BYTE,
    onSuccess: (uniqueUploadID, optionalArguments) => {
        UnityEngine.Debug.Log("Tiledmedia log upload success. Unique upload id: " + uniqueUploadID);
    },
    onFailure: (clearVRMessage, optionalArguments) => {
        UnityEngine.Debug.Log("Tiledmedia log upload failed. Error details: " + clearVRMessage);
    }
);
```

Upgrading to v9

This section describes the upgrade cycle from v8.x to v9.x. If your project is still using an older SDK, please refer to the Upgrading to v8.x section first.

New API design

> **NOTE**
>
> While this document is written against the ClearVR SDK for Unity, the API changes also apply to the ClearVR SDK for Native Android and -iOS.

We have changed the design of the asynchronous APIs of all platforms to be:

- Safer to use
- More fitting to the platform.

Single `Action<>` based callbacks on ClearVRPlayer APIs (like `Seek()`, `Pause()` and `SwitchContent()` were introduced in `v4.3-277-gbc48e2d4 (2018-12-14)`. These callbacks allow one to directly relate one request (e.g. "Seek") with a specific response (e.g. "Seek completed OK") instead of relying on e.g. the related `StateChanged` events. This is especially helpful if many requests are processed in quick succession. For backwards compatiblity reasons, pre v9.x SDKs contained equivalent APIs that did not require you to specify a callback. In v9.x, these backwards compatible APIs have been forcefully deprecated and can no longer be used.

These changes will result in easier and cleaner app level code. We replaced all these APIs with equivalents on all platforms with the same name as you already know.

An example of how you can implement these new APIs in Unity:

```
clearVRPlayer.controller.Pause(
    onSuccess: (argClearVREvent, argClearVRPlayer) => { /* handle success */ },
    onFailure: (argClearVREvent, argClearVRPlayer) => { /* handle failure */ }
);
```

These work in a fairly simmilar matter for Native Android and -iOS. For more examples of the implementation in Unity or for examples on Native Android and Native iOS please see the included sample applications.

Mesh positioning

Mesh positioning has been overhauled in v9.0. In prior version, there was just a single video mesh at any point in time. Now, with the support for mosaic playback. there can be any number of meshes

(now called Display Objects) in one scene. All v9.x SDKs support a backwards compatible legacy mode where the old automatic mesh placement mechanism is mimicked. This mode, however, will be removed in v10.0. Please refer to the (Layout Manager)[layoutmanager.md] documentation for important details and examples.

Deprecated APIs and functionality

The following methods and fields have been removed:

MediaPlayer

- `ClearVRAsyncRequest ParseMediaInfo(String argUrl);`
- `void ParseMediaInfo(String argUrl, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `void ParseMediaInfo(ContentItem argContentItem, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `ClearVRAsyncRequest PrepareContentForPlayout(ContentItem argContentItem);`
- `void PrepareContentForPlayout(ContentItem argContentItem, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `void Recenter();`
- `void Rotate(Vector3 argNewEulerAngles);`
- `void Translate(Vector3 argTranslation, Space argRelativeTo = Space.Self);`
- `void Translate(Vector3 argTranslation, Transform argRelativeTo);`
- `bool SetMainColor(Color argNewColor);`
- `ClearVRAsyncRequest PrewarmCache(String argManifestUrl, int argInitialPositionInMilliseconds, long argFlags);`
- `ClearVRAsyncRequest PrewarmCache(String argManifestUrl, long argInitialPositionInMilliseconds, long argFlags);`
- `void PrewarmCache(String argManifestUrl, int argInitialPositionInMilliseconds, long argFlags, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `FallbackLayout GetFallbackLayout();`
- `Texture2D GetTexture();`
- `void HideOrUnhide(bool argHideOrUnhide);`
- `ClearVRAsyncRequest SetStereoMode(bool argStereo);`
- `GameObject GetActiveClearVRMesh();`
- `ClearVRDisplayObjectController GetActiveClearVRDisplayObjectController();`
- `bool GetIsMediaInfoParsed();`

Controller

- `ClearVRAsyncRequest Stop();`
- `ClearVRAsyncRequest Pause();`
- `ClearVRAsyncRequest Unpause();`
- `ClearVRAsyncRequest StartPlayout();`
- `ClearVRAsyncRequest TogglePause();`

- `void SetAudioTrack(int argIndex, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `ClearVRAsyncRequest SetAudioTrack(int argIndex);`
- `void SetAudioTrack(int argIndex, AudioDecoder argAudioDecoder, AudioPlaybackEngine argAudioPlaybackEngine, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `ClearVRAsyncRequest SetAudioTrack(int argIndex, AudioDecoder argAudioDecoder, AudioPlaybackEngine argAudioPlaybackEngine);`
- `void Stop(bool argStoppedAfterApplicationLostFocus, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `ClearVRAsyncRequest Stop(bool argStoppedAfterApplicationLostFocus);`
- `ClearVRAsyncRequest Seek(long argNewPositionInMilliseconds, long argFlags = (long) SeekFlags.None);`
- `void Seek(long argNewPositionInMilliseconds, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived = null, params object[] argOptionalArguments);`
- `void Seek(long argNewPositionInMilliseconds, long argFlags, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `ClearVRAsyncRequest SwitchContent(ContentItem argContentItem);`
- `ClearVRAsyncRequest SwitchContent(ContentItem argContentItem, AudioDecoder argAudioDecoder, AudioPlaybackEngine argAudioPlaybackEngine);`
- `void SwitchContent(ContentItem argContentItem, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `void SwitchContent(ContentItem argContentItem, AudioDecoder argAudioDecoder, AudioPlaybackEngine argAudioPlaybackEngine, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);ClearVRAsyncRequest`
- `SwitchContent(ContentItem argContentItem, long argFlags);`
- `void SwitchContent(ContentItem argContentItem, long argFlags, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `void SwitchContent(ContentItem argContentItem, AudioDecoder argAudioDecoder, AudioPlaybackEngine argAudioPlaybackEngine, long argFlags, Action<ClearVREvent, ClearVRPlayer> argCbClearVRAsyncRequestResponseReceived, params object[] argOptionalArguments);`
- `ClearVRAsyncRequest SwitchContent(ContentItem argContentItem, AudioDecoder argAudioDecoder, AudioPlaybackEngine argAudioPlaybackEngine, long argFlags);`
- `long GetCurrentContentTimeInMilliseconds();`
- `long GetCurrentWallclockContentTimeInMilliseconds();`

MediaInfo

- `String GetHighestQualityResolutionAndFramerate();`
- `String GetCurrentResolutionAndFramerate();`
- NativeSDK-Android and NativeSDK-iOS only: `int getDecoderVideoFrameWidth()` and `int getDecoderVideoFrameHeight()`
- `long GetContentDurationInMilliseconds();`

ContentItem

- `int DEFAULT_HFOV_OFFSET_360_CONTENT`
- `int hfovOffset`
- `long startPositionInMilliseconds`

- `float startYaw`
- `float start`
- `long approximateDistanceFromLiveEdgeInMilliseconds`
- `ContentItem(String argManifestUrl, ClearVRViewportAndObjectPose argStartViewportAndObjectPose = null, long argStartPositionInMilliseconds = 0, ContentFormat argOverrideContentFormat = ContentFormat.Unknown, DRMInfo argDRMInfo = null, FishEyeSettings argFishEyeSettings = null, long argApproximateDistanceFromLiveEdgeInMilliseconds = 0);`
- `ContentItem(String argManifestUrl, long argStartPositionInMilliseconds, float argStartYaw, float argStartPitch = 0f, int argHfovOffset = 0);`
- `ContentItem();`
- `ContentItem Clone(long argStartPositionInMilliseconds);`

The following APIs have been removed from the `ClearVRMeshBase` (renamed to `ClearVRDisplayObjectController` in v9.x) class:

- `void Rotate(Vector3 argNewEulerAngles);`
- `void Translate(Vector3 argTranslation, Space argRelativeTo = Space.Self);`
- `void Translate(Vector3 argTranslation, Transform argRelativeTo);`
- `void Recenter();`
- `void ResetPose(ContentItem argContentItem);`
- `Pose GetDefaultDisplayObjectPose();`
- `void HideOrUnhide;` -> Set the parent gameObject `activeSelf` flag instead.
- `String GetStatusAsString();` -> Use `ToString()` instead.
- `FallbackLayout GetFallbackLayout();`

The following fields have been removed from the `ClearVRDisplayObjectController` object:

- `eulerAngles`
- `position`
- `containerGameObject`
- `isVisible`

The following fields have been deprecated from `PlatformOptions` and can no longer be referenced:

- `sphereLayer`
- `sphereRadius`
- `isVRDevicePresent` -> refer to `Utils.GetIsVRDevicePresent()` instead.

The following constructors have been deprecated and can no longer be used:

- `PrepareContentParameters(argContentItem, argTimingParameters = null, argSyncSettings = null, argFlags = 0, argTimeoutInMilliseconds = 0, argApproximateDistanceFromLiveEdgeInMilliseconds = 0, argLayoutParameters = null)`
  . Use the `PrepareContentParameters(ContentItem, TimingParameters, LayoutParameters)` constructor instead and set the other, optional, fields manually.

- `SwitchContentParameters(argContentItem, argTimingParameters = null, argTransitionType = TransitionTypes.Fast, argAudioTrackAndPlaybackParameters = null, argSyncSettings = null, argFlags = 0, argApproximateDistanceFromLiveEdgeInMilliseconds = 0, argPreferredAudioTrackLanguage = "")`. Use the `SwitchContentParameters(ContentItem, TimingParameters, LayoutParameters)` constructor instead and set the other, optional, fields manually.

ClearVREvent-related changes

`ClearVREventTypes.StateChangedStopping` is no longer emitted when platformOptions.applicationFocusAndPauseHandling is set to its default valueRecommended and the application has lost focus. Instead, the new SuspendingPlaybackBeforeApplicationPaused event is triggered in this case.

Dumping verbose ClearVR logs

Sometimes, we ask you to temporarily enable verbose logging when you run into an unforeseen problem. This is controlled by setting:

```
ClearVRPlayer.coreLogLevel = 2;
ClearVRPlayer.coreLogFile = "/storage/emulated/0/clearvrcore.log";
```

On recent versions of Android (Android 10+), writing directly to the root of the device's storage (`/storage/emulated/0/`) is no longer allowed (see: https://developer.android.com/about/versions/11/privacy/storage). The following code will work on all Android versions and all platforms:

```
ClearVRPlayer.coreLogLevel = 2;
ClearVRPlayer.coreLogFile = Application.persistentDataPath + "/clearvrcore.log";
```

Refer to https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html to see where the log files can be retrieved.

iOS (Unity and Native SDK):

Version 9.0 of the ClearVR SDK adds support for more architectures and platforms, like the arm64 and x86_64 emulator as well as AppleTV. This support is provided via `Assets/ClearVR/Plugins/iOS/MediaFlow.xcframework` and requires iOS 13 or newer. For backwards compatibility reasons, we also provide the `Assets/ClearVR/PLugins/iOS/MediaFlow.framework` framework that requires iOS 11 or newer but does only work on real (non-simulator) arm64 iOS devices. A post-process script will make sure that the most suitable variant is included in your exported project.

⚠ WARNING

When importing a v9.x SDK on top of an older SDK, the old and new `framework` might clash with one-another. If

you run into build failures related to the (xc) framework, please remove all files from `Assets/ClearVR/Plugins/iOS` and re-import your latest unity package.

v8.0 and older SDKs came with a `MediaFlow.framework`. v9.0 and newer includes both a `MediaFlow.framework` and an `xcframework`. The former requires iOS 11 and up, while the latter requires iOS 13+ but brings support for other platforms like simulator and AppleTV.

- In Unity: the post process script will automatically select the appropriate `framework` or `xcframework` depending on the iOS version you target.
- In Native apps: remove the `MediaFlow.framework` folder from your project and include the new `MediaFlow.xcframework` instead as an Embedded library.

Upgrading to v8

The v8.0 major release sees the removal of a number of (long time) deprecated APIs, but also a number of fundamental changes that will require changes to your code. They are listed here.

> **NOTE**
>
> This document only applies to developers who have not yet migrated to v8.0.

> **NOTE**
>
> While this document is written against the ClearVR SDK for Unity, all changes also apply to the ClearVR SDK for Native Android and -iOS. All APIs are virtually identical across platforms, with the most notable exception being that `platformOptions` in the Unity SDK is equal to `ClearVRPlayerParameters` in the Native SDKs.

In short

Quick overview of changed/removed APIs and enums, see the subsequent sections for details:

- `SeekFlags` has been renamed to `TimingTypes`
- `SeekFlags.None` has been renamed to `TimingTypes.ContentTime`.
- The new `TimingParameters` object brings initial playback start-, seek- and switch content timing together.
- `contentItem.startPositionInMilliseconds` has been replaced with `TimingParameters`.
- Previously, When calling `Seek(0)` you would seek to the Live Edge in a live stream regardless the SeekFlags you might have specified. Now you MUST set `TimingTypes.LiveEdge`, e.g.: `Seek(new TimingParameters(0, TimingTypes.LiveEdge))`
- `SeekParameters(long, SeekFlags)` constructor has been changed to `SeekParameters(TimingParameters(), TransitionTypes)`.
- `platformOptions.autoPrepareContentItem` has been moved to `platformOptions.prepareContentParameters(contentItem, TimingParameters)`
- `SwitchContentParameters` constructor has changed.
- `GetTimingReport(SeekFlags)` has been renamed to `GetTimingReport(TimingTypes)`.
- The TimingReport now contains the duration as well.
- `PrewarmCacheFlags` has been removed.
- `ClearVRPlayer.mediaPlayer.prewarmCache()` has been removed, no replacement will be added. Please remove any reference from your code.
- `ClearVREventTypes.PrewarmCacheCompleted` has been removed. Please remove any reference from your code.
- (rarely used) `ClearVRPlayer.controller.Stop(bool, callback)` has been replaced by `ClearVRPlayer.controller.Stop(callback)`.

ContentItem

- The `ContentItem.startPositionInMilliseconds` field has been removed. The `ContentItem` is now considered to be a "static" object during its lifetime which is created once and should not see any changes throughout its lifetime. Previously, you would've updated the `startPositionInMilliseconds` field on an existing ContentItem when switching content, but you can now achieve the same with a much more logical API. The new `TimingParameters` object has been added to control the start position when loading the first clip, when switching content and when seeking.

TimingParameters

This new object aggregates a timing position in milliseconds and a `TimingTypes` which defines how to interpret the provided value (`TimingTypes` was formerly known as `SeekFlags`). You will be creating this object when loading the first `ContentItem`, switching content to a different `ContentItem` using `SwitchContent()` and when you call `Seek`. The `TimingTypes` defines whether the position should be interpreted as `ContentTime`, `WallclockTime`, `RelativeTime`, `LiveEdge` and the new `Seamless` type. Please refer to the detailed Scripting API documentation for more information on these `TimingTypes`, and remember that not all `TimingTypes` are supported by all APIs.

Specify first ContentItem and its start position.

When initializing the player, you typically configure it by specifying various `platformOptions`. The `ContentItem` to load during player initialization was set by configuring `platformOptions. autoPrepareContentItem` and the start position was set when calling the constructor on the `ContentItem` (or you relied on it being set to the default value: 0 msec). In v8.0, things have moved around a bit.

1. The start position is no longer configured when constructing the `ContentItem` (remember that it was implicitly set to 0 if not explicitly set in your call on the constructor).
2. The `autoPrepareContentItem` field has been removed from `platformOptions` and has been replaced by the more versatile `platformOptions.prepareContentParameters` field, which takes a `PrepareContentParameters(ContentItem, TimingParameters, ...)` object.
3. To specify the start position in milliseconds (and how to interpret that start position), one has to create a `TimingParameters(long, TimingTypes)` object.

Making the changes above is sufficient to migrate to this new v8 API.

Seeking

Specifying your `SeekParameters` when seeking has slightly changed and now leverages the new `TimingParameters` object described above. Pre-v8.0, you would construct a

`SeekParameters(long, SeekFlags)` object, this is now replaced by the `SeekParameters(TimingParameters, TransitionTypes)` constructor. The `TimingParameters` object has been desrcibed in the previous section. The `TransitionTypes` argument is for future use and has no effect yet. You are recommended to keep it at its default value `TransitionTypes.Fast`.

> **⚠ WARNING**
>
> Previously, When calling `Seek(0)` you would seek to the Live Edge in a live stream regardless the `SeekFlags` you might have specified. Now you MUST set `TimingTypes.LiveEdge`, e.g.:
> `Seek(new TimingParameters(0, TimingTypes.LiveEdge))`.
> `Seek(new TimingParameters(0, TimingTypes.ContentTime))` will seek to the earliest available content (which might not start at position 0 depending on the cache window on the Content Delivery Network).

Switching content

Pre-v8.0, switching content consisted of updating/creating a `ContentItem` with the preferred `startPositionInMilliseconds`, creating a `SwitchContentParameters` object and calling `SwitchContent()`. In v8.0, you no longer need to construct/update your `ContentItem`. Instead, you construct a `SwitchContentParameters(ContentItem, TimingParameters, TransitionTypes)` object where the `TimingParameters` object specifies the start position and how this start position should be interpreted (as discussed above). Furthermore, since v8.0, you can now specify how the *transition* between the two `ContentItem`s should "look like". This argument defaults to `TransitionTypes.Fast`, which is equal to the behaviour you are accustomed to (e.g. playback briefly halts while the switch is performed as fast as possible). v8.0 introduces the ability to switch seamless between two `ContentItem`. Provided that both `ContentItem` are frame-accurately in sync, you will get a perfect smooth switch-over. To achieve this, you have to set the timingParameters to `TimingParameters(0, TimingTypes.Seamless)` and the transitionType to `TransitionTypes.Continuous`. Remember that the accuracy and perceived "quality" of the seamless switch depends on network conditions and how accurately the audio and video track in both `ContentItem`s were synchronized during production.

Dumping verbose ClearVR logs

Sometimes, we ask you to temporarily enable verbose logging when you run into an unforeseen problem. Previously, this was controlled by adding/enabling the following code in your project:

```
platformOptions.clearVRCoreVerbosity = 2;
platformOptions.clearVRCoreLogToFile = "/storage/emulated/0/clearvrcore.log";
```

These two fields have been removed and can no longer be set. Instead, you should set the following two static fields `ClearVRPlayer.coreLogLevel` and `ClearVRPlayer.coreLogFile` respectively. The log settings will take effect when calling `clearVRPlayer.Initialize()` or `static ClearVRPlayer.TestIsContentSupported()`, *whichever is called first*. Changing them after calling

either of these two APIs will have no effect.

> **⚠ WARNING**
>
> Setting non-default log settings has a negative performance impact. Make sure to always set both fields to their default values (`0` and an empty string respectively) at all times.