

# Components

- CVR Asset Info
- CVR Avatar
- CVR World
- CVR Spawnable
- CVR Distance Constraint
- CVR Pickup Object
- CVR Pointer
- CVR Advanced Avatar Settings Pointer
- CVR Toggle State Pointer
- CVR Parameter Stream

# CVR Asset Info

Required component that is automatically added alongside **CVR Avatar** / CVR World / CVR Spawnable.

**"Request failed. The provided content ID does not belong to your account."**

You may run into this error if you've manually added the CVR Asset Info component to your asset!

Remove both the CVR Asset Info script and your CVR Avatar / CVR World / CVR Spawnable component.

Then readd only the CVR Avatar / CVR World / CVR Spawnable component and try again.

## Unique Identifier

This field is used to identify your content when uploading. You can attach an existing Guid belonging to your account for updating existing content, or leave it empty to automatically generate a new Guid on upload.

# CVR Avatar

The main component for a ChilloutVR avatar. **This is required for an avatar upload.**

Some options listed below may be missing for generic (non-humanoid) avatars.

This component will automatically add the required [CVR Asset Info](#) component for you. You may run into issues uploading if you've manually added the CVR Asset Info script before the CVR Avatar component!

## Generic Avatar Settings

Property	Description
View Position	Controls your avatars viewpoint in-game. This should be between both eyes.
Voice Position	Controls your avatars voice position in-game. This should be on your avatars mouth.
Voice Parent	The humanoid bone which your Voice Position is attached to. It will follow the bone's position while you move your avatar around in world space. Head, Left Hand, Right Hand, Hips.

## Avatar Customization

### Animation Overrides

The Override Controller used to specify which Controller and Overrides to use on the avatar.

You can use this to quickly override animations on the default Controller included with the CCK, or alternatively reference your own custom Controller to use on the avatar instead.

## Blinking and Visemes

You can configure your avatar to have eye movement, blinking and voice activated blendshapes called visemes.

### Face Mesh

The Skinned Mesh Renderer to use for automatic blinking and visemes.

## Use Eye Movement

Enables automatic eye movement that can focus on nearby players.

This requires eye bones to be mapped in your avatars Humanoid configuration!

This property is animatable during runtime.

## Use Blink Blendshapes

Enables automatic blinking using the specified blendshapes. You are able to specify up to four blendshapes to use at once.

Any unneeded blendshapes can be left as .

This property is animatable during runtime.

## Use Lip Sync

Enables lip sync to be used on your avatar. You can choose between using Visemes, Single Blendshape, and Jaw Bone.

This property is animatable during runtime. It must be enabled by default when using the Visemes Lip Sync Mode.

## Lip Sync Mode

- Visemes
  - Uses multiple blendshapes to map human speech to mouth movement.  
You can find examples and references from the [Oculus Developer Viseme Reference Documentation](#).
- Single Blendshape
  - Uses a single blendshape to drive mouth movement.
- Jaw bone
  - Uses the mapped jaw bone in your avatars Humanoid configuration to drive mouth movement.

## Viseme Smoothing

A value between 1-100 to determine the smoothing of the Visemes Lip Sync Mode.

Lower values will snap to the currently recognized Viseme, and higher values will slowly smooth to it.

# Advanced Tagging

With the advanced tagging feature, you can set content filter tags for specific gameobjects on your avatar.

Press the  button to add an entry to your **Tagged Gameobjects** list.

## Tags

List of content tags you want for this particular gameobject. You can tag Loud Audio, Long Range Audio, Screen Fx, Flashing Colors, Flashing Lights, Violence, Gore, and Horror.

## GameObject

The specific gameobject you would like to add the above tags to.

## Fallback GO (GameObject)

A fallback gameobject to use instead if the above gameobject is hidden by the content filter.

If you are using the Advanced Tagging System, you still need to Tag your Avatar appropriately on the upload screen!

# Advanced Settings

# CVR World

The main component for a ChilloutVR world. **This is required for a world upload.**

# CVR Spawnable

The main component for a ChilloutVR prop. **This is required for a prop upload.**

# CVR Distance Constraint

This component acts like a tether, keeping an object within a certain distance of a reference object.

## **Target**

This is the transform of the reference object. It serves as an anchor point for the tether.

## **Min Distance**

This is the minimum distance, in meters, to keep from the reference object. Defaults to zero (0).

## **Max Distance**

This is the maximum distance, in meters, to keep from the reference object. Defaults to zero (0).

## **Current Distance**

This is the current distance from the reference object.



# CVR Pickup Object

This script allows an object to be picked up by a user.

## Grip Type

This controls how a user will hold the object

### Free

This mode specifies that the object can be grabbed anywhere, and will be held wherever the user grabs it.

### Origin

This mode specifies that the object should be held in a specific orientation relative to the user's hand.

## Grip Origin

This is the transform that is used to indicate where the object should be held. A gizmo will show a left-handed grip, around the transform.

The grip transform is arranged Z+ (blue arrow) forward and Y+ (green arrow) up.

If you require a separate origin for Desktop players, create a child GameObject on your Grip Origin, and name it `[ Desktop ]`. This transform uses Y+ (green arrow) forward, and Z+ (blue arrow) down.

## Disallow Theft

Checking this box disallows other users from taking this item while it is currently being held.

## Maximum Grab Distance

This is the maximum distance the user can be from the object and still interact with it. The setting defaults to 0, but 2-3 is usually a good starting point.

## Snapping References

*(This section requires more research.)*

# Auto Hold

This allows the object to stay in the hand indefinitely without requiring the user to actively hold onto it.

Desktop users can press G in order to release the held object.

VR users can hold Grip and down on the joystick to release the held object.

# IK Reference

*(This section requires more research.)*

# CVR Pointer

A component used to physically interact with the many different variations of triggers.

As the name suggests, this component marks a **point** that will activate triggers once they've come into contact.

## Type

The string used to identify the pointer. Allows a trigger to specify what types of pointers should interact.

CVR Pointer will check for any existing colliders before adding its own Sphere Collider during runtime.

This allows you to change the pointer size or shape by adding your own Collider(s) with IsTrigger checked.

## Trigger Types

There are different types of triggers that a pointer can interact with:

### Avatar

- CVR Advanced Avatar Trigger
- CVR Toggle State Trigger

### World

- CVR Interactable

### Spawnable

- CVR Interactable
- CVR Spawnable Trigger

# CVR Advanced Avatar Settings Pointer

This component should be considered deprecated. You can use [CVR Pointer](#) instead.

CVR Advanced Avatar Settings Pointer inherits from CVR Pointer internally, and as such will act the same.

# CVR Toggle State Pointer

This component should be considered deprecated. You can use [CVR Pointer](#) instead.

CVR Toggle State Pointer inherits from CVR Pointer internally, and as such will act the same.

# CVR Parameter Stream

This component allows you to use a multitude of local variables as inputs for Animators, Variable Buffers, Avatar Parameters, and Spawnable parameters.

As of writing, this component is only officially supported on Avatars to set Avatar Parameters, but does function as intended on Spawnables and in Worlds.

This script is underdeveloped in its current state and likely will need a full rework in the future. It is incredibly powerful and functions, but note that some aspects of this script are easily abusable or not fully thought out.

## Note on Update Rate

The script will update entries every 0.05s for performance reasons, so depending on your usecase you may notice values are choppy and need smoothing via Animator shinanigans.

## Parameter Name

The string of the parameter you want to apply the value to.

## Target

Depending on the selected **TargetType**, this GameObject is used to get the Animator or Variable Buffer components to set the outputted value to.

## Type

The local variable used to drive the selected Animator Parameter.

Some Type entries are not fully developed, non-functional, or poorly implemented.

- TimeSeconds
  - DateTime.Now returned as Float.
- TimeSecondsUtc
  - DateTime.UtcNow returned as Float.
- DeviceMode
  - If in VR returns 1, otherwise returns 0.
- HeadsetOnHead

- If in VR & Headset is on head returns 1, otherwise returns 0.

**This does not function at the time of writing.**

- ZoomFactor

- Returns the current Desktop Camera Zoom value between 0 and 1.  
This value should be direct 0-1 based on if not zoomed or fully zoomed.

- ZoomFactorCurve

- Returns the current Desktop Camera Zoom Curve value between 0 and 1.  
This value should be smoother than the above type when zooming.

- EyeMovementLeftX

- Returns eyeAngle.x from CVREyeController.

**This is currently bugged & poorly implemented.**

- EyeMovementLeftY

- Returns eyeAngle.x from CVREyeController.

**This is currently bugged & poorly implemented.**

- EyeMovementRightX

- Returns eyeAngle.x from CVREyeController.

**This is currently bugged & poorly implemented.**

- EyeMovementRightY

- Returns eyeAngle.x from CVREyeController.

**This is currently bugged & poorly implemented.**

- EyeBlinkingLeft

- Returns blinkProgress from CVREyeController.

**This returns the total blinking progress and is not independent of the other eye.**

- EyeBlinkingRight

- Returns blinkProgress from CVREyeController.

**This returns the total blinking progress and is not independent of the other eye.**

- VisemeLevel

- Returns visemeLoudness from CVRVisemeController.

**VisemeLevel is constantly adjusting to your volume level.** Consistent voice volume will be pretty accurate, with visemeLoudness being scaled to your average maximum volume. Sudden loud noises will cause VisemeLevel to become less sensitive until it readjusts to your average volume.

- TimeSinceHeadsetRemoved

- Returns the amount of time you've had your headset off for.

**This does not function at the time of writing.**

- TimeSinceLocalAvatarLoaded

- Returns the time since you've last switched into an avatar.

- LocalWorldDownloadPercentage

- Returns the current normalized download percentage while loading a World.

**Note that once the world has finished downloading & started loading the value will return to 0.**

- LocalFPS

- Returns your current FPS as a float.

- LocalPing

- Returns your current Ping as a float.

- LocalPlayerCount
  - Returns the current Player Count in the World as a float.  
**Can never hit 0 as it always adds 1 to account for the local player.**
- LocalTimeSinceFirstWorldJoin
  - Time since you've loaded into your Home World after launching the game.  
**Can be used to read how long you've been in-game.**
- LocalTimeSinceWorldJoin
  - Time since you've loaded into any World.  
**Can be used to read how long you've been in a World.**
- LocalPlayerMuted
  - Returns 1 if Muted, 0 if Unmuted.
- LocalPlayerHudEnabled
  - Returns 1 if Hud is enabled, and 0 if disabled.
- LocalPlayerNameplatesEnabled
  - Returns 1 if Nameplates are enabled, and 0 if disabled.  
**Will still return 1 if Nameplates are set to Menu Only.**
- LocalPlayerHeight
  - Returns the Player Height setting in General Settings as a float.  
**This value doesn't return exact Player Height.** Internally this returns (  $(\text{int})\text{PlayerHeight} / 100f / 1.084f$  ).  
 You will likely need to play around with this value to tune it for your usecase.
- LocalPlayerControllerType
  - **Not implemented.**
- LocalPlayerFullBodyEnabled
  - Returns 1 if FBT, 0 if otherwise.  
**This returns 1 if Full Body Tracking is AVAILABLE, not CALIBRATED.**
- TriggerLeftValue
  - Returns the Left Trigger value between 0 and 1.
- TriggerRightValue
  - Returns the Right Trigger value between 0 and 1.
- GripLeftValue
  - Returns the Left Grip value between 0 and 1.
- GripRightValue
  - Returns the Right Grip value between 0 and 1.
- GrippableObjectLeft
  - **Not implemented.**
- GrippableObjectRight
  - **Not implemented.**
- TransformGlobalPositionX
  - **Not implemented.**
- TransformGlobalPositionY
  - **Not implemented.**
- TransformGlobalPositionZ
  - **Not implemented.**
- TransformGlobalRotationX
  - **Not implemented.**
- TransformGlobalRotationY



- **Not implemented.**
- TransformGlobalRotationZ
  - **Not implemented.**
- TransformLocalPositionX
  - **Not implemented.**
- TransformLocalPositionY
  - **Not implemented.**
- TransformLocalPositionZ
  - **Not implemented.**
- TransformLocalRotationX
  - **Not implemented.**
- TransformLocalRotationY
  - **Not implemented.**

## TargetType

The target you want to use the output for.

Avatar Animator is currently the only supported & selectable TargetType as defined in CCK3.4, but internally CVRParameterStream supports these other TargetTypes and are currently functional in-game.

- Animator
  - Uses **Target** GameObject property to get Animator component.
  - Output will always be a Float.
- Variable Buffer
  - Uses **Target** GameObject property to get Variable Buffer component.
  - Output will always be a Float.
- Avatar Animator
  - Sets parameters onto the **Local Player Avatar**.
  - Output will be cast from Float to the correct Parameter Type.

**Anything above 0.5f will return true if casting to Bool or Trigger.**
- Custom Float
  - Sets parameters onto the found **Spawnable** component.
  - Output will always be a Float.

CVRParameterStream runs locally for every player, so keep in mind Network Sync when utilizing on Spawnables via the Custom Float TargetType. It may be best to use the Animator TargetType instead to avoid the Spawnable's ownership constantly changing as every client attempts to sync parameters to the parent Spawnable.

Nothing prevents you from using Avatar Animator TargetType on Spawnables or Worlds, but the game will now refuse to set parameters if the target CVRAvatar is not your local CVRAvatar component. (this was abusable for a period of time)

# Static Value

A float value used to adjust values based on the selected **Application Type**.

## Application Type

Allows you to choose from a list of preset arithmetic to adjust the returned values.

For explanation sake, we will define a few variables:

**StaticValue** = the **Static Value** Float set above.

**CurrentValue** = the current **Avatar Animator** parameter value as a Float.

**(This is always 0 for all other TargetTypes.)**

**ReturnedValue** = the returned value from the above **Type** entry.

**FinalValue** = the final value after the **Application Type** arithmetic.

- Override
  - Sets returned value directly with no adjustment.  
**FinalValue = ReturnedValue;**
- AddToCurrent
  - Adds returned value to current animator value.  
**FinalValue = CurrentValue + ReturnedValue;**
- AddToStatic
  - Adds static value to returned value.  
**FinalValue = StaticValue + ReturnedValue;**
- SubtractFromCurrent
  - Subtracts returned value from current animator value.  
◦ **FinalValue = ReturnedValue - CurrentValue;**
- SubtractFromStatic
  - Subtracts returned value from static value.  
◦ **FinalValue = StaticValue - ReturnedValue;**
- SubtractWithCurrent
  - Subtracts returned value from current animator value.  
◦ **FinalValue = CurrentValue - ReturnedValue;**
- SubtractWithStatic
  - Subtracts static value from returned value.  
◦ **FinalValue = ReturnedValue - StaticValue;**
- MultiplyWithCurrent
  - Multiplies the returned value with current value.  
◦ **FinalValue = ReturnedValue \* CurrentValue;**
- MultiplyWithStatic
  - Multiplies returned value with static value.  
◦ **FinalValue = ReturnedValue \* StaticValue;**
- CompareLessThen
  - Returns 1 if returned value is less than static value, 0 if otherwise.  
◦ **FinalValue = ((ReturnedValue < StaticValue) ? 1f : 0f);**
- CompareLessThenEquals

- Returns 1 if returned value is less than or equal to static value, 0 if otherwise.
  - **FinalValue = ((ReturnedValue <= StaticValue) ? 1f : 0f);**
- CompareEquals
  - Returns 1 if returned value is equal with static value, 0 if otherwise.
  - **FinalValue = ((ReturnedValue == StaticValue) ? 1f : 0f);**
- CompareMoreThenEquals
  - Returns 1 if returned value is more than or equal to static value, 0 if otherwise.
  - **FinalValue = ((ReturnedValue >= StaticValue) ? 1f : 0f);**
- CompareMoreThen
  - Returns 1 if returned value is more than static value, 0 if otherwise.
  - **FinalValue = ((ReturnedValue > StaticValue) ? 1f : 0f);**
- Mod
  - **FinalValue = ReturnedValue % Mathf.Max(Mathf.Abs(StaticValue), 0.0001f);**
- Pow
  - **FinalValue = Mathf.Pow(ReturnedValue , StaticValue);**