Volume Proxies Quick Start

A quick guide on getting started with Audio Gameplay Volume Proxies.

> Learn to use this **Beta** feature, but use caution when shipping with it.

You can use **Volume Proxy Components** to interact with the **Audio Gameplay Volume** system without a traditional Audio Gameplay Volume object.

This guide will teach you how to create dynamic volumes using both Proxy types: **Primitive** and **Condition**.

Prerequisites

- Create a new First Person Template project.
- Enable the Audio Gameplay Volumes plugin from the **Plugin** panel by selecting **Edit > Plugins**, using the search bar to find the plugin, and then clicking the corresponding checkbox.

1 - Set Up an Ambient Volume Sound Class

Create and configure a new **Sound Class** for the project to support Audio Gameplay Volumes.

1. Create a new Sound Class.
    a. In the **Content Browser**, click the **Add** button.
    b. Select **Audio > Classes > Sound Class**.
    c. Give the newly created Asset a name, such as `AGV_SoundClass`.
2. Double-click the Sound Class to open the **Sound Class Editor**.
3. In the **Details** panel:
    a. Enable **Routing > Apply Ambient Volumes**.
    b. Set **Submix > Default 2DReverb Send Amount** to 1.0, then save it.
4. Set the Sound Class as the Project default. This will assign it to all sound sources without a specified Sound Class override.
    a. Open the **Project Settings** by clicking **Edit > Project Settings** from the top menu bar.
    b. Enter "Default Sound Class" into the search bar.
    c. Click the dropdown beside **Audio > Default Sound Class** and select your new Sound Class.

2 - Create a Blueprint Actor with a Reverb Component

Create a new Blueprint Actor with a Reverb Component as a base for the Proxy Components you will add in following steps.

> ⓘ You can add Proxy Components to **any** Blueprint Actor. Audio Gameplay Volume Components attached to that Proxy Actor will function as if they were on a traditional Audio Gameplay Volume object.

1. Create a new Blueprint Actor for the Audio Gameplay Volume Proxy.
    a. In the **Content Browser**, click the **Add** button.
    b. Select **Blueprint Class**.
    c. From the **Pick Parent Class** window, select **Actor**.
    d. Give the newly created Asset a name, such as `AGV_Proxy`.
2. Double-click the Proxy Blueprint Class in the **Content Browser** to open the **Blueprint Editor**.

3. Add a Reverb Component.
    a. In the **Components** panel, click the **Add** button.
    b. Type "Reverb" into the search bar and press Enter.
4. Select the new Reverb Component (named `ReverbVolume` by default).
5. Adjust the Reverb Component's properties in the **Details** panel.
    a. Set **Reverb > Volume** to 1.0.
    b. Set **Reverb > Fade Time** to 0.0.
    c. Click the dropdown beside **Reverb > Reverb Effect**. Under the Create New Asset heading, select **Reverb Effect**.
    d. Give the new Reverb Effect a name, such as `AGV_Reverb`, and save it.
6. Modify the Reverb Effect.
    a. Double-click the icon beside **Reverb > Reverb Effect** to open the **Reverb Effect Editor**.
    b. Set **Late Reflections > Gain** to 1.0 to make the effect easier to hear.
    c. Save the Reverb Effect and close the Reverb Effect Editor.

3 - Construct a Primitive Proxy

A **Primitive Proxy** uses a primitive Static Mesh on the Actor as the basis for collision to trigger enter and exit events.

3A - Add the Primitive Proxy Component

Add the base Volume Proxy Component and then set its type to Primitive.

1. Add a Volume Proxy Component.
    a. In the **Components** panel, click the **Add** button.
    b. Type "Volume Proxy" into the search bar and press Enter.
    c. Name the Component `PrimitiveProxy`.
2. Select the PrimitiveProxy Component.
3. In the **Details** panel, set **Audio Gameplay > Proxy** to AGV Primitive Proxy.

3B - Add the Static Mesh Component

Add a Static Mesh and configure its collision so you can enter and exit it.

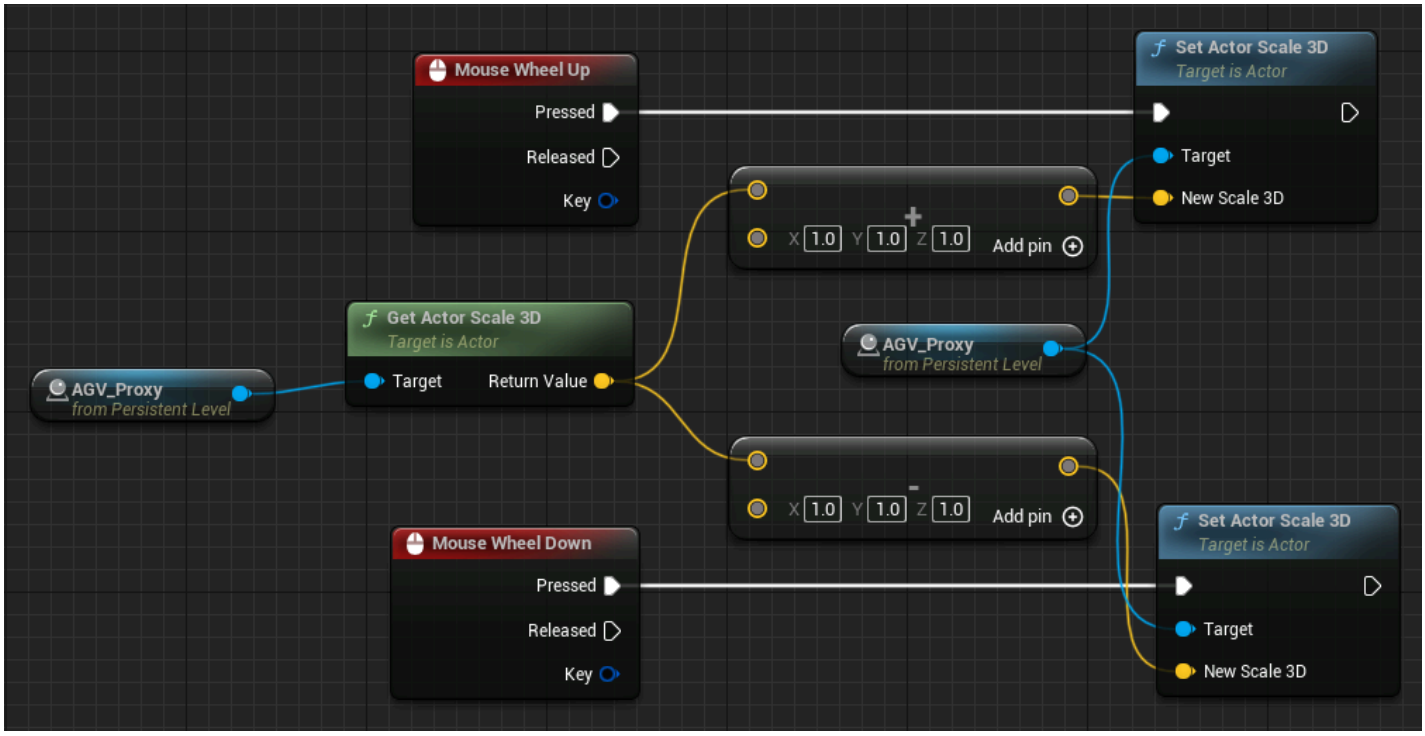
> Primitive Proxies do not currently support Actors with multiple Static Mesh components. A warning will appear in the Output log if you use such an Actor.

1. Add any Static Mesh Component.
    a. In the **Component** panel, click the **Add** button.
    b. Type "Static Mesh" into the search bar and select one of the Basic Shapes from the list, such as Sphere.
2. Select the Static Mesh Component.
3. Set the Collision parameters in the **Details** panel.
    a. Disable **Collision > Generate Overlap Events**.
    b. Set **Collision > Can Character Step Up On** to No.
    c. Set **Collision > Presets** to Custom.
    d. Under **Collision > Collision Presets > Collision Responses**, select **Overlap**. This selects Overlap for all response types.
4. Save the Blueprint.

3C - Place the Proxy Actor into the Level

1. Place the Proxy Actor by dragging it from the **Content Browser** into the **Level**.
2. Using the transformation widget, place the Actor so your character can move in and out of it.

3D - Build the Level Blueprint



Build the Blueprint Graph to control the scale of the Proxy Actor with mouse wheel input.

1. Select the Proxy Actor in the Level.
2. On the **Level Editor Toolbar**, click the **Blueprint** button and select **Open Level Blueprint**.
3. Right-click in an empty space and **Create a Reference** to the Proxy Actor.
4. On the **Actor Reference** node, drag off the output pin and create a **Get Actor Scale 3D** node.
5. On the **Get Actor Scale 3D** node, drag off the **Return Value** pin and create an **Add** and a **Subtract** node.
6. On both the **Add** and **Subtract** nodes, enter 1.0 in the X, Y, and Z of the bottom inputs.
7. On the **Actor Reference** node, drag off the output pin and create two **Set Actor Scale 3D** nodes.
8. Connect the output of the **Add** node to one of the **Set Actor Scale 3D** nodes.
9. On the same **Set Actor Scale 3D** node, drag off the **Exec Input (>)** pin and create a **Mouse Wheel Up** node.
10. Connect the output of the **Subtract** node to the other **Set Actor Scale 3D** node.
11. On the same **Set Actor Scale 3D** node, drag off the **Exec Input (>)** pin and create a **Mouse Wheel Down** node.

3E - Test the Level

Click the **Play** button on the **Level Editor Toolbar**, pick up the rifle by moving into it, and shoot it by left-clicking. You can alter the scale of the Proxy using the mouse wheel.

Move in and out of the Audio Gameplay Volume Proxy and notice how it affects the rifle firing sound. The reverb effect still applies within a dynamically resized Proxy.

4 - Construct a Condition Proxy

A **Condition Proxy** triggers the enter and exit events based on a condition specified in Blueprint through the **Audio Gameplay Condition** interface.

4A - Add the Condition Proxy Component and Interface

Add the base Volume Proxy Component and then set its type to Condition.

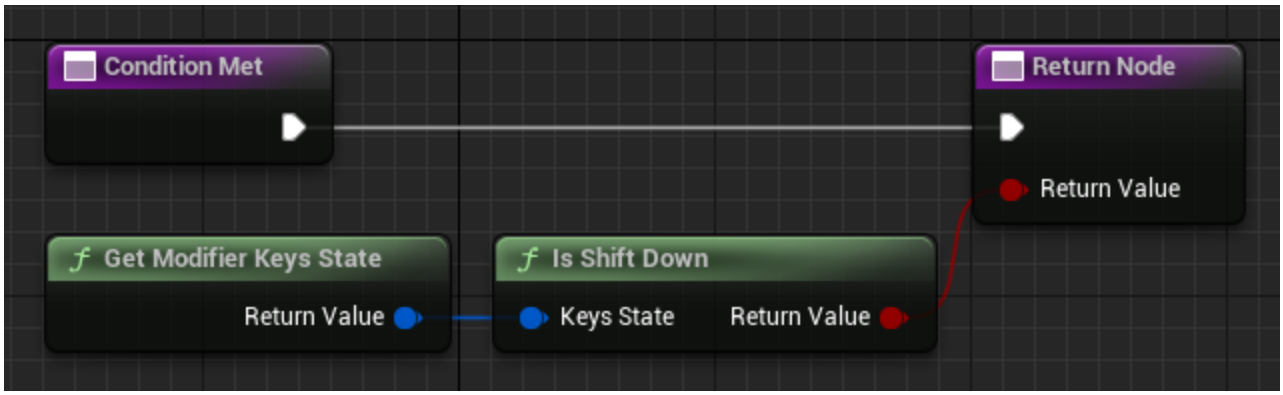
1. Double-click the Proxy Blueprint Class in the **Content Browser** to open the **Blueprint Editor**.
2. Add a Volume Proxy Component.
    a. In the **Components** panel, click the **Add** button.
    b. Type "Volume Proxy" into the search bar and press Enter.
    c. Name the Component `ConditionProxy`.
3. Select the ConditionProxy Component.
4. In the **Details** panel, set **Audio Gameplay > Proxy** to AGV Condition Proxy.

4B - Add the Audio Gameplay Condition Interface

Add the Audio Gameplay Condition interface so you can build the conditions in Blueprint. This adds two new interfaces, **Condition Met** and **Condition Met Position**.

1. On the **Blueprint Editor Toolbar**, click **Class Settings**.
2. Click the **Add** dropdown for **Interfaces > Implemented Interfaces**.
3. Type "AudioGameplayCondition" into the search bar and press Enter.

4C - Build the Condition Met Interface

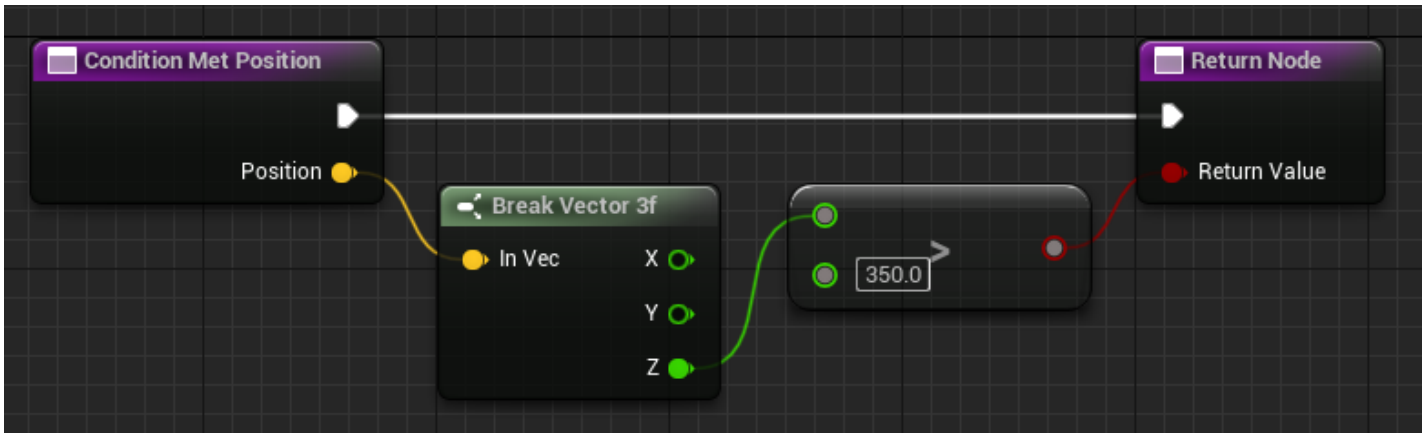


The **Condition Met** interface is used to set up any condition that results in a boolean value. With it, you can make your own custom criteria for what defines being inside or outside of the volume.

Construct a shift key state check in Blueprint to drive the **Condition Met** interface.

1. In the **My Blueprint** panel, under **Interfaces > Audio Gameplay Condition**, double-click the **Condition Met** interface.
2. On the **Return Node**, drag off the **Return Value** input pin and create an **Is Shift Down** node.
3. On the **Is Shift Down** node, drag off the **Keys State** input pin and create a **Get Modifier Keys State** node.

4D - Build the Condition Met Position Interface



The **Condition Met Position** interface is similar to the Condition Met interface but gives you access to the listener's position for use within your custom condition.

Construct a listener height check in Blueprint to drive the **Condition Met Position** interface.

1. In the **My Blueprint** panel, under **Interfaces > Audio Gameplay Condition**, double-click the **Condition Met Position** interface.
2. On the **Condition Met Position** node, drag off the **Position** output pin and create a **Break Vector 3f** node.
3. On the **Break Vector 3f** node, drag off the **Z** output pin and create a **Greater** node.
4. On the **Greater** node:
    a. Enter 350.0 in the bottom input.
    b. Connect the output pin to the **Return Node**.
5. Save the Blueprint.

4E - Test the Level

Click the **Play** button on the **Level Editor Toolbar** and test the rifle firing sound. The reverb effect from the Proxy applies when holding Shift or when shooting on top of a platform.

5 - On Your Own!

Now that you've finished creating some basic Volume Proxies, consider taking it even further. Below are a couple of suggestions you can explore on your own.

- You've added custom triggers, but you can also add custom responses using the **On Proxy Enter** and **On Proxy Exit** event nodes in Blueprint. To create one, select the target Volume Proxy Component for the event, right-click in an empty space on the **Event Graph**, and add the **On Proxy** event node.
- By default, the camera is considered to be the listener and is used to determine enter and exit events. With a third-person camera, you may want the listener to be the Player Character instead. Try using a Condition Proxy to override the default volume collision in the Third Person Template project.