Wwise Unreal Integration
Requirements

Unreal Engine

Each release of the Unreal Wwise plug-in is customized for certain versions of the Unreal Engine. Be sure to use one of the supported versions of the Unreal Engine, as indicated in What's New?

The latest supported Unreal version for this plug-in is Unreal Engine 4.18.
Wwise

This plug-in is based on Wwise 2017.2.1 build 6524.; while it is possible to use an alternate version of the Wwise SDK, modifications to the plug-in code may be necessary.
Platforms

This plug-in has been tested on Windows, Mac, Linux (SteamOS), Xbox One, PlayStation 4, Android, and iOS. Modifications may be required to target other platforms.
Development Environment Setup

Please refer to the UE4 documentation:

- "Downloading Unreal Engine Source Code"
- "Setting Up Visual Studio for UE4"
- "Building Unreal Engine from Source"
Wwise Unreal Integration
Installation

Before proceeding, review the Requirements page and make sure the appropriate versions of the Unreal Engine, Unreal Wwise plug-in, and Wwise are used.

This UE4 integration package contains files related to the Wwise plug-in only; it does not contain the entire Unreal Engine source code, nor the Wwise SDK.
Installation Procedure

There are two ways of installing the Unreal Wwise plug-in: either as a game plug-in or as an engine plug-in - not both.

**Note:**
For more information on the difference between engine and game plug-ins (sometimes referred to as "installed" plug-ins), please refer to the Unreal Engine Wiki.

**Installing the Wwise Unreal Plug-in as a Game Plug-in**

Installing as a game plug-in is handled by the Wwise Launcher. Please open the Launcher to the UNREAL ENGINE tab and find your Unreal project. If it's not there, then click Browse for project... to find the UPROJECT file and add the project to the Launcher. Once it's listed, click Integrate Wwise into Project... for the Unreal project. The Launcher will guide you to specify all the necessary information to successfully integrate the Wwise plug-in into your project. Alternatively, you can accomplish the same thing using offline integration files, which are downloaded via the UNREAL ENGINE tab menu.

**Managing Wwise Plug-ins**

There are various plug-ins available for Wwise, which need to be installed correctly in order to be used in the Unreal Wwise plug-in. Plug-ins for Wwise are managed via the Wwise Launcher. For more information on managing plug-ins for Wwise, refer to the PLUG-INS section of the Installation and Migration Guide.

**Note:**
If there are plug-ins missing from your Wwise installation when you use the Wwise Unreal plug-in, you will see the following error: LogAkAudio: Error: Could not find plugin dynamic library.

**Installing the Wwise Unreal Plug-in as an Engine Plug-in**
Note:
Installing the Wwise plug-in for Unreal as an engine plug-in is intended for expert users only.

To install as an engine plug-in, you must first download the integration from the Launcher. Go to the UNREAL ENGINE tab and select the Download Wwise Integration to use as an engine plug-in... option from the tab's menu.

Then, follow these steps:

1. Copy the "Wwise" UE4 Integration folder to the "Plugins" folder. Copy the "Wwise" UE4 Integration folder to …/<UE4 installation directory>/Engine/Plugins.
2. Since the Wwise UE4 integration may need to be rebuilt on the fly by the Unreal Editor during the packaging process, several folders from the Wwise SDK installation folder must be copied into the "Wwise" plug-in folder hierarchy. The following folders are required:
   - Source: C:\Program Files (x86)\Audiokinetic\Wwise 2017.2.1 build 6524.\SDK\include\*.*
   - Destination: .../Plugins/Wwise/ThirdParty/include
   - Source: C:\Program Files (x86)\Audiokinetic\Wwise 2017.2.1 build 6524.\SDK\Win32_*\*.*
   - Destination: .../Plugins/Wwise/ThirdParty/Win32_*
   - Source: C:\Program Files (x86)\Audiokinetic\Wwise 2017.2.1 build 6524.\SDK\x64_*\*.*
   - Destination: .../Plugins/Wwise/ThirdParty/x64_*
3. You then need to open your project and go to **Edit > Plugins > Audio** and select the **Enable** check box under "Wwise Unreal Engine 4 integration".

For Android, iOS, Linux, PlayStation 4, and Xbox One, the libraries must also be copied:

- **Source**: `C:\Program Files (x86)\Audiokinetic\Wwise 2017.2.1 build 6524.\SDK\<Your Platform>\*.star`  
- **Destination**: `.../Plugins/Wwise/ThirdParty/<Your Platform>`
Building the plug-in

Preprocessor Defines

The following preprocessor defines are introduced to control the integration features:

- **AK_USE_UNREAL_IO**
  Located in ...
  \Plugins\Wwise\Source\AkAudio\AkAudio.Build.cs.
  This setting, which is enabled by default, causes all IO requests from
  the Wwise sound engine to be routed in the Unreal IO system. When
  not enabled, the standard Wwise low-level I/O is used.

- **AK_SOUNDFRAME**
  Located in ...
  \Plugins\Wwise\Source\AkAudio\AkAudio.Build.cs.
  This setting provides basic SoundFrame support to activate Unreal
  Editor features, such as Radius display, that require communication
  with the Wwise Authoring Application. Enabled by default, except in
  the Ship configuration. This option is not available on Mac OS X.
Building the Unreal Wwise plug-in from source

If the installation steps outlined in Installation have been followed correctly, rebuilding the Unreal Wwise plug-in is done the same way as rebuilding the Unreal Engine. For a more detailed description of this process, please consult the Unreal Engine documentation.

For more information on plugins with source code, please refer to the Unreal Engine documentation.
Wwise Unreal Integration
Using the Integration

- **Initial Setup** How to set up using Wwise after building Unreal with the integration.

- **Workflow**
  A basic workflow to get started.

- **UE4 C++ projects**
  A quick overview to get started using a C++ project.

- **Sample Game**
  A quick overview of the UnrealWwiseDemo game demonstrating the integration features.

- **Features**
  Survey of the features introduced by the integration.

- **Frequently Asked Questions**
  Answers to the most frequently asked questions.
Initial Setup

Initialization settings

**Note:** All settings related to the Wwise Integration can be found in the Unreal Project Settings, under the Wwise section of the plug-ins section.

- **Max Simultaneous Reverb Volumes**
  The maximum number of Ak Reverb Volumes that can affect a sound. Setting this to zero disables all Ak Reverb Volumes in game.

- **Wwise Windows Installation Path**
  The location of the Wwise Authoring Application on your Windows development machine. This option will need to be updated when a new version of the Wwise Authoring Application is required by the integration changes.

- **Wwise Mac Installation Path**
  The location of the Wwise Authoring Application on your Mac OS X development machine. This option will need to be updated when a new version of the Wwise Authoring Application is required by the integration changes.

- **Wwise Project Path**
  The location of the Wwise project for the UE4 game. The Wwise integration requires this path to create the Wwise assets required for the game using the Generate SoundBanks function within the Unreal Content Browser (or Build menu). This path is relative to the editor's executable directory (usually `<UE4_installation_directory>/Engine/Binaries/Win64`), as given by FPlatformProcess::BaseDir() in the Unreal Engine.

If these paths are not correctly set, Unreal will fail to generate the Wwise SoundBanks required for the game.
Initializing the SoundEngine

The SoundEngine initialization steps are performed in the FAKAudioDevice::EnsureInitialized() method. In this method, the memory, streaming, IO, sound engine, platform, music engine, and communication settings can be configured.

Plug-ins can be registered by including their associated plug-in factory header file in

`.../Plugins/Wwise/Source/AkAudio/Private/AkAudioDevice.cpp` under the line saying: // Add additional plug-ins here.

Linking against a plug-in's library can be done by adding a call to the AddWwiseLib() function for the plug-in in

`.../Plugins/Wwise/Source/AkAudio/AkAudio.Build.cs`, near the other plug-ins. Once this is complete, the UE4 game project will have to be rebuilt from its Visual Studio solution or Xcode workspace.

Please note that a limitation in Unreal Engine 4 prevents rebuilding an Unreal plug-in from a Content-only (Blueprint) project. Therefore, third-party Wwise DSP plug-ins are not available in a Content-only (Blueprint) project, as this method requires rebuilding the Wwise Unreal plug-in.

For more information on SoundEngine initialization, refer to the Initialize the Different Modules of the Sound Engine section in the Wwise SDK documentation.
Workflow

Here is a sample workflow using the integration:
1) Create a Wwise project

Create a Wwise project for this game using the Wwise Authoring Application. Some Events are created, but no SoundBanks need to be created yet.
2) Add AkAudioEvent objects to Unreal packages

In the Unreal Content Browser, add AkAudioEvent objects to packages.
3) Reference AkAudioEvent objects in a level

AkAudioEvent objects can now be referenced in a level, such as in using Wwise-specific Blueprint Functions or AkAmbientSound objects.
4) Add AkAudioBank objects

The Wwise assets associated with AkAudioEvent object can be distributed across any number of AkAudioBank objects. Using the Unreal Content Browser, add AkAudioBank objects as required.
5) Set the RequiredBank property for each AkAudioEvent

The RequiredBank property of each AkAudioEvent determines which AkAudioBank its content will be stored in. This property must be set for each AkAudioEvent used in the game.
6) Generate SoundBanks (for play-in-editor)

From the Build menu, select the Generate SoundBanks option, and then select all banks to be generated, and click OK. The required SoundBanks are generated. After the banks are generated at least once, it is possible to play sounds in the editor.

Note: The first time banks are generated, you need to click Refresh All Banks by right-clicking on one of the AkAudioBank objects.

Banks generated from the Unreal Editor can then be added to source control. This allow individuals without the Wwise application and project on their machine to still be able to work in the Unreal editor using existing BNK files from the game’s Content folder. They could hook the AkAudioEvent objects at different places in the level, and they would be able to run the game.

When new banks are generated, click Refresh All Banks in the Content Browser's AkAudioBank to load the newly created banks.
UE4 C++ projects

Linking with the AkAudio Module

The AkAudio module must be linked with to use Wwise's functionality within a C++ project. This must be done within your project's module file (.build.cs).

For example:

```csharp
public class MyModule : ModuleRules
{
    public MyModule(ReadOnlyTargetRules Target) : base(Target)
    {
        PublicDependencyModuleNames.AddRange(
            new string[]
        { // Other settings
        });
    }
}
```

**Note:** All AK::SoundEngine calls are currently made from AkAudioDevice and **must** be made from within the AkAudio module. Users must extend the functionality within the AkAudio module to expose required Wwise functionality.
Sample Game

A sample game containing the Wwise UE4 Integration is available to download in the Wwise Launcher. Please refer to the Launcher.

The game's map provides several demonstration stations.
Generation of the SoundBanks

Note: Generate SoundBanks for this project from the WwiseDemoGame Unreal Editor.

1. Open the WwiseDemoGame in the Unreal Editor.
2. From the toolbar, click the arrow next to Build and select Generate SoundBanks... from the Audiokinetic category.
3. Select all the SoundBanks and the desired platforms and click OK.

On completion, the resulting content is visible in ...
.../WwiseDemoGame/Content/WwiseAudio.
Map Contents

AkEvent Animation Notify

An Animation Notify can be used to post AkEvents to the SoundEngine. A demonstration of its use can be seen in the FPP_Fire animation.

Ambient Demo

This area of the map demonstrates the use of the AkAmbientSound actor. The AmbientNoise_Spatialized ambient emitter in the level is started by a Start All Ambient Sounds node in the Level Blueprint. To help locate the actor, its attenuation sphere is drawn in yellow when playing the game.

The Event used by this emitter is part of the AmbientBank SoundBank, which is automatically loaded thanks to the Auto Load flag set on its AkAudioBank asset.

This area also helps to demonstrate the occlusion feature. By navigating the player pawn behind the wall situated close to the ambient sound (making sure to stay within the yellow zone), or behind the big box in the sphere, the effects of occlusion on the sound can be heard. Occlusion is enabled on the sound by setting it in the in the "Ambient Sound handling" (red) section of the Blueprint level.

Sequencer Demo

This section of the map demonstrates the use of Event and RTPC tracks in the WwiseDemoSequence asset. Opening the level sequence in the Sequencer editor shows that the Event track plays a drum beat targeting the moving cube, while the DrumKitModulation game parameter is driven by the RTPC track.

RTPC Demo

This section of the map demonstrates use of the Set RTPCValue node
targeting an actor, in the Level Blueprint (in the green comment sections). The mouse scroll wheel (the up and down D-pad buttons on a gamepad, and a two-finger swipe on a touch screen) is tied to a variable that is fed as the "Velocity" Game Parameter, which controls the pitch of the VelocityLoop sound in the Wwise project.

The "Create RTPC button Event dispatchers" section of the level Blueprint also shows how to manually load and unload the VelocityBank (whose Auto-Load is unselected), using the **Load Bank** and **Unload Bank** Blueprint nodes. It also demonstrates how to post an Event that targets an actor, using the **Post Event** node.

**Reverb Demo**

This section of the map demonstrates use of AkReverbVolume. Inside the spherical cave, an AkReverbVolume is used to add a Reverb Effect to the weapon sound. Note that the actor emitting the sound can be set to ignore the reverb volumes feature. This is done in the "Variable initializations" section of the Blueprint level.

**Switch Demo**

This section of the map demonstrates how to set a Switch using blueprints (in the "Switch Logic" section of the Blueprint level). Using the button (see instructions on the wall in front of the button) posts the Event to the SoundEngine. Pressing L or H on the keyboard (D-Pad left and right on a gamepad, three and four finger taps on a touch screen) changes the switch value.

**WAAPI Widgets Demo**

This section of the map demonstrates how to use the **WAAPI UMG Widgets** to control Wwise directly from your game.

The demo on the left wall illustrates how to use Unreal Engine's built-in widgets along with Blueprint scripting to control Wwise. The keyboard in step 1 changes the text in the Editable Text Box. When that text is changed, a WAAPI Call is made using **WAAPI Blueprint Functions** to search for items in Wwise matching the entered text (as can be seen in
step 2), as is shown in the "Search for an item" section in the Widget's Graph. WAAPI returns the list of matching items in a **JSON objects**, which is then parsed to populate the combo box found in step 3. Once an item is selected, it is possible to play or stop it using the buttons. The Blueprint script to control this behaviour is found in the "Play/Stop" the selected item section of the Widget's Graph.

**Note:**
Due to an Unreal Engine known issue on the Mac platform, updating ComboBox values in a 3D widget does not work. You can still experience the demo using the 2D version of these widgets, available by pressing "P" in game.

The demo on the back wall demonstrates how the **Ak Wwise Tree** and **Ak Item Bool Properties** widgets can be used to control a **Ak CheckBox**. The tree can be used to find a Wwise object to control. That item can be dragged & dropped onto the UE Checkbox or the Ak Checkbox to set the item the checkbox controls. The same has to be done with the Bool Properties picker. Once the item and the property have been set on the checkboxes, you may use them to enable or disable the property directly in Wwise. Changing the property in Wwise will also be reflected on both checkboxes. The UE Checkbox showcases how you can use the built-in Checkbox widget and Blueprint scripting to accomplish this. The Ak Checkbox shows that the exact same behaviour can be accomplished without any Blueprint scripting.

The demo on the right wall is very similar to the previous Checkbox demo, but shows how to use a **Ak Wwise Tree** and a **Ak Item Properties** to control a **Ak Slider**.

Note that all three demos are also available in normal 2d UMG widgets by pressing the "P" key on your keyboard during gameplay.
Features

The following sections give an overview of the elements that the integration brings to Unreal.

- **Unreal Objects** Unreal Objects introduced to expose base Wwise concepts.

- **Editor Integration**
  WAAPI-enabled workflow enhancements to the Editor.

- **Blueprint Functions**
  Blueprint functions available in visual scripting.

- **Animation**
  Animation Notify for Events.

- **Level Sequencer**
  Sequencer Tracks for Events and RTPC.

- **Matinee**
  Matinee Tracks for Events and RTPC.

- **Occlusion**
  Occlusion support.

- **Wwise Authoring API (WAAPI)**
  Wwise Authoring API (WAAPI)
Unreal Objects

Spatial Audio Objects WAAPI Widgets
AkAudioEvent

Unreal object representing a Wwise Event. Can be created either by drag-and-drop from the Wwise Picker or by right-clicking in the Unreal Content Browser. Its name should exactly match the name of an Event in the Wwise Project.

- Properties
  - **Required Bank**: Specifies the SoundBank that contains this Event along with its structures and media.
- Unreal Content Browser Context Menu Options
  - **Play Event**: Posts the Event.
  - **Stop Event**: Stops all currently playing Events.
AkAuxBus

Unreal object representing a Wwise Auxiliary Bus. Can be created either by drag-and-drop from the Wwise Picker, or by right-clicking in the Unreal Content Browser. Its name should exactly match the name of an Auxiliary Bus in the Wwise Project. This allows to add plug-in media to a SoundBank from within the Unreal Editor.

- Properties
  - **Required Bank**: Specifies the SoundBank that contains this Auxiliary Bus along with its associated plug-in media.
AkAudioBank

Unreal object representing a Wwise SoundBank. Can be created by right-clicking in the Unreal Content Browser.

- Properties
  - **Auto Load**: When enabled, automatically loads and unloads the SoundBank along with the package referencing it. **Note**: Enabling this flag does not instantly load the bank in the editor; it is only applied on the next package load.

- Unreal Content Browser Context Menu Options
  - **Generate Selected SoundBank(s)...**: Launches a dialog window where a SoundBank Generation operation can be performed on a list of SoundBanks.
  - **Load Bank**: Loads this SoundBank.
  - **Unload Bank**: Unloads this SoundBank.
  - **Clear Banks**: Unloads all currently loaded SoundBanks, including the Init bank.
  - **Load Init Bank**: Loads the Init Bank. This Bank must be loaded prior to any other SoundBank.
  - **Refresh All Banks**: Stops all sounds; unloads and then loads all currently loaded SoundBanks.
AkAmbientSound

AkAmbientSound is an AActor class used in the same way as the AAmbientSound object that is supplied with the default Unreal Audio system. Its playback is controlled either via its own object Blueprint functions, or by using the global helper functions Start All Ambient Sounds and Stop All Ambient Sounds. An AkAmbientSound also contains an AkComponent, which has its own properties.

- Properties
  - **Stop When Owner Is Destroyed**: When enabled, the Event is stopped automatically when the AkAmbientSound is destroyed.
  - **Auto Post**: Automatically post the associated AkAudioEvent on BeginPlay.
- Blueprint Functions
  - **Start All Ambient Sound**: Starts the playback of all ambient sounds.
  - **Start Ambient Sound**: Starts the playback of the selected ambient sound.
  - **Stop All Ambient Sound**: Stops the playback of all ambient sounds.
  - **Stop Ambient Sound**: Stops the playback of the selected ambient sound.
AkReverbVolume

AkReverbVolume is an AVolume class used in a similar way to the AREverbVolume object that is supplied with the default Unreal Audio system. It can be spawned from any Brush in the Editor. The reverb effect is obtained via an AkLateReverbComponent.
AkLateReverbComponent

This component can be added to any volume, allowing to create a reverb zone from it. The reverb effect is obtained by assigning a Wwise Auxiliary Bus to the component, and routing all AkComponents entering this volume to the associated Wwise Auxiliary Bus. If there is volume overlap, a Priority property is used to determine which Auxiliary Buses the target AkComponent is routed to. A temporal fade in/out Effect is applied to the level of the Auxiliary Bus when entering/exiting a AkReverbVolume.

- Properties
  - Enable Late Reverb: Enables or disables this component.
  - Aux Bus: AkAuxBus assigned to this volume. This aux bus should enable game-defined auxiliary sends. If you are using Late Reverb with AkRoomComponent and AkAcousticPortal, it also needs to enable positioning set to 3D.
  - Send Level: Maximum Send Level associated with the Wwise Auxiliary Bus.
  - Fade Rate: Rate at which to fade in/out the SendLevel of the current Late Reverb Component when entering/exiting it, in percentage per second (0.2 will make the fade time 5 seconds).
  - Priority: The precedence in which the Late Reverb Components will be applied. In the case of overlapping volumes, only the ones with the highest priority are chosen (the number of simultaneous Late Reverb Components is configurable in the Unreal Editor Project Settings under Plugins > Wwise). If two or more overlapping Late Reverb Components have the same priority, the chosen Late Reverb Components is unpredictable.
AkComponent

AkComponent is derived from USceneComponent and represents an active Wwise event.

- Properties
  - **Attenuation Scaling Factor**: If the Ambient Sound uses 3D attenuation in Wwise, this property allows modifying the attenuation computations on this ambient sound to simulate sounds with a larger or smaller area of effect.
  - **Occlusion Refresh Interval**:
  - **Ak Audio Event**: The AkAudioEvent that is posted when the AkAmbientSound object is instructed to start playing. If you want to use Spatial Audio features, the sound effect of the event should enable game-defined auxiliary sends.
  - **Spatial Audio Properties**:
    - **Early Reflection Aux Bus**: AkAuxBus with the AkReflect plug-in. Set to None to disable geometric reflections. This aux bus should enable game-defined auxiliary sends with enable positioning set to 2D.
    - **Early Reflection Aux Bus Name**: If no AkAuxBus is entered in Early Reflection Aux Bus, Early Reflection Aux Bus Name will be used. Set to AK_INVALID_UNIQUE_ID to disable geometric reflections.
    - **Early Reflection Order**: Maximum number of reflections that will be processed when computing indirect paths via the geometric reflections API. Reflection processing grows exponentially with the order of reflections, so this number should be kept low. Valid range: 1-4.
    - **Early Reflection Bus Send Gain**: Send gain (0.f-1.f) that is applied when sending to the bus that has the AkReflect plug-in.
    - **Early Reflection Max Path Length**: A heuristic to stop the computation of reflections. Should be no longer (and possibly shorter for less CPU usage) than the maximum attenuation of the sound emitter.
    - **Enable Spot Reflectors**: Enable reflections on
**AkSpotReflector.**

- **Debug Draw** options: This allows you to visualize the raycasts performed by the Spatial Audio Engine, as well as the triangles hit by these raycasts, allowing you to easily debug what is going on in the Spatial Audio Engine. This only works on one component at a time.

- **Blueprint Functions**
  - **Post Ak Event**: Starts playback of the specified Event.
  - **Post Trigger**: Posts a Trigger to the Event associated with the component.
  - **Set Listeners**: Sets the listeners of this UAkComponent.
  - **Set Occlusion Refresh Interval**: Sets the time interval at which the UAkComponent performs occlusion calculations. Set to 0 to turn off occlusion on the component.
  - **Set RTPC Value**: Sets the value of the RTPC of the Event associated with the component.
  - **Set Stop when Owner Destroyed**: Sets the StopWhenOwnerDestroyed value on the component.
  - **Set Switch**: Sets a Switch of the Event associated with the component.
  - **Stop**: Stops playback of the Event associated with the component.
  - **Use Reverb Volumes**: Sets whether the component is influenced by AAkReverbVolumes or not.
Wwise Unreal Integration » Using the Integration » Features » Unreal Objects
Spatial Audio Objects

AkAcousticTexture

Unreal object representing a Wwise Acoustic Texture. Can be created either by drag-and-drop from the Wwise Picker or by right-clicking in the Unreal Content Browser. Once created, it can be applied on a AkSurfaceReflectorSetComponent's polygons or on a AkSpotReflector.

- Properties
  - **Edit Color**: Editor-only property defining the color to be used when colorizing AkSurfaceReflectorSetComponent polygons that have an AkAcousticTexture assigned.
AkSpotReflector

The Spot Reflector actor is an experimental feature. It allows to place an omnidirectional point reflector in the 3D world. Spot reflectors make sense when they are placed on far away objects that have a large radius; like a distant mountain.

Spot Reflectors will reflect any sound coming from an **AkComponent** that has **Enable Spot Reflectors** enabled and that is positioned in the same area. An **AkComponent** will feed to spot reflectors if they both are outside rooms. If the **AkComponent** is in a room, it will only feed to spot reflectors inside the same room. Rooms can be created with a volume that has an **AkRoomComponent** attached.

Calls **AK::SpatialAudio::AddImageSource()** from the Spatial Audio API on **BeginPlay**.

- **Properties:**
  - **Aux Bus**: **AkAuxBus** that has the AkReflect plug-in for early reflection DSP. This aux bus should enable game-defined auxiliary sends with enable positioning set to 2D.
  - **Acoustic Texture**: **AkAcousticTexture** used to filter sounds reflected by this image source.
  - **Distance Scaling Factor**: Image source distance scaling. This number effectively scales the **sourcePosition** vector with respect to the listener and, consequently, scales distance and preserves orientation.
  - **Level**: Game-controlled linear level for this source.
AkSurfaceReflectorSetComponent

Representation of a Geometry Set in Unreal Engine 4. This component needs to be attached on a AVolume actor. On BeginPlay, all of the volume’s enabled polygons will be sent to the SpatialAudio engine.

- **Properties**
  - **Enable Surface Reflector Set**: Enables or disables this component
  - **Acoustic Surfaces**: Array defining the AkAcousticTexture associated with each polygon. The index in the array is represented in the Game Viewport when the parent volume is selected. Once a texture is assigned to a polygon, the polygon is colorized using the texture’s Edit Color, and the Acoustic Texture’s name is printed on the polygon in the Game Viewport. Assigning None as an acoustic texture will make the surface completely reflectible, no additional texture filter will be applied. Each polygon can also be enabled or disabled with the Enable Surface checkbox.
**AkRoomComponent**

This component can be added to any volume, and then creates a Spatial Audio Room. Rooms have two purposes:

- Allow for oriented reverb. All auxiliary buses applied on a game object within a room will be oriented towards the volume's front vector.
- Used in conjunction with **AkAcousticPortal**, allows routing the wet signal from one room through a portal, to another room.

In both cases, the auxiliary bus used should have its positioning options set to 3D, with positioning enabled, and have an attenuation assigned.

- **Properties**
  - **Enable Room**: Enables or disables this component
  - **Priority**: The precedence in which the rooms will be applied. In the case of overlapping rooms, the one with the highest priority is chosen. If two or more overlapping rooms have the same priority, the chosen room is unpredictable.
  - **Wall Occlusion**: Used to set the Wwise occlusion value on emitters in the Room, when no audio paths to the listener are found via sound propagation in Wwise Spatial Audio. This value can be thought of as 'thickness', because it relates to how much sound energy is transmitted through the wall. The valid range is 0.0f-1.0f, and is mapped to the occlusion curve as defined in the Wwise project.
AkSpatialAudioVolume

For your convenience, an AkSpatialAudio volume is included in the integration. It consists of a simple volume with a
AkSurfaceReflectorSetComponent, a AkRoomComponent and a AkLateReverbComponent attached.
AkAcousticPortal

Representation of a Portal in Unreal Engine 4. Allows for sounds contained in a volume with AkRoomComponent attached to leak into other volumes with AkRoomComponent attached. Such volumes overlapping with Portals are detected at initialization time. Both 2D and 3D sounds can be routed through a portal.

- Properties
  - Initial state: Whether this Portal should be initialized in an open (enabled) or closed (disabled) state.
  - Obstruction Refresh Interval: Time interval between occlusion/obstruction checks. Set to 0 to disable occlusion on this component.
  - Obstruction Collision Channel: The object collision channel used for creating the collision of the obstruction ray-casts (between the Portal and the listener) with the desired geometry.
Wwise Unreal Integration » Using the Integration » Features » Unreal Objects
WAAPI Widgets

WAAPI UMG Widgets

Ak Check Box

This WAAP-enabled check box allows you to control a boolean property in Wwise.

Properties:

- **Property to control**: Name of the boolean property to control in Wwise.
- **Item to control**: Item to control in Wwise.
- **On Check State Changed**: Event called when the check box’s state changes.
- **On Item Dropped**: Event called when an item is dropped on the check box.
- **On Property Dropped**: Event called when a property name is dropped on the check box.

Blueprint methods:

- **IsPressed**: Returns true if this button is currently pressed.
- **IsChecked**: Returns true if the check box is currently checked.
- **GetCheckedState**: Returns the full current checked state.
- **SetIsChecked**: Sets the checked state.
- **SetCheckedState(ECheckBoxState InCheckedState)**: Sets the checked state.
- **SetAkItemId**: Sets the item ID this check box controls.
- **GetAkItemId**: Returns the ID of the currently controlled item.
- **SetAkBoolProperty**: Sets the property name this check box controls.
- **GetAkProperty**: Returns the current item property.
**Ak Item Bool Properties**

Pre-made picker for Bool properties. Useful for drag & drop use with Ak Check Box or Ak Slider.

**Ak Item Properties**

Pre-made picker for float properties. Useful for drag & drop use with Ak Check Box or Ak Slider.

**Ak Slider**

This WAAPI-enabled slider is able to control any float property in Wwise. When the controlled property is changed, the range for the slider's values will automatically be adjusted to the same range as in Wwise.

Properties:

- **Property to control**: Name of the boolean property to control in Wwise
- **Item to control**: Item to control in Wwise
- **Value**: The slider's value
- **On Value Changed**: Event called when the slider's value changes
- **On Item Dropped**: Event called when an item is dropped on the slider
- **On Property Dropped**: Event called when a property name is dropped on the slider

Blueprint methods:

- **GetValue**: Returns the current value.
- **SetValue**: Sets the current value.
- **SetCheckedState(ECheckBoxState InCheckedState)**: Sets the checked state.
- **SetAkSliderItemId**: Sets the item ID this slider controls.
- **GetAkSliderItemId**: Returns the ID of the item currently controlled.
- **SetAkSliderItemProperty**: Sets the property name this slider controls.
- **GetAkSliderItemProperty**: Returns the property name this slider controls.

**Ak Wwise Tree**

This widget is an exact copy of The WAAPI Picker, available in-game. It allows to assign items to Ak Check Box and Ak Slider via drag & drop.

![Ak Wwise Tree](image)

**Properties:**

- **On Selection Changed**: Event called when the tree's current selection changes.
- **On Item Dragged**: Event called when an item is dragged from the tree.

Blueprint methods:

- **GetSelectedItem**: Returns the item currently selected in the tree.
- **GetSearchText**: Returns the current text in the search box.
- **SetSearchText**: Sets the current text of the search box.

**Ak Wwise Tree Selector**

More compact version of the Ak Wwise Tree, which pops a separate window to select the item.

Properties:

- **On Selection Changed**: Event called when the tree's current selection changes.
- **On Item Dragged**: Event called when an item is dragged from the tree.
Slate Widgets

**FWwiseTreeItem**

Structure containing everything needed to represent a Wwise item.

Members:

- **DisplayName**: The item's name
- **FolderPath**: The path of the tree item in Wwise, including the name
- **ItemType**: The type of item
- **ItemId**: The ID of the item

**FWwisePropertyDragDropOp**

Handles drag & drop of a property name. Simply pass the property name to `FWwisePropertyDragDropOp::New` to create the drag & drop operation.

**FWwiseUmgDragDropOp**

Handles drag & drop of a Wwise item. Simply pass a shared pointer to a `FWwiseTreeItem` to `FWwisePropertyDragDropOp::New`, in order to create the drag & drop operation.

**Ak Slider**

This WAAPi-enabled slider is able to control any float property in Wwise. Is able to handle drag & drop via `FWwisePropertyDragDropOp` and `FWwiseUmgDragDropOp`.

Public methods:

- **GetAkSliderValue**: Gets the current value of the slider, as displayed in the Wwise Authoring application.
- **GetAkSliderProperty**: Gets the name of the controlled property.
- **GetAkSliderItemControlled**: Gets the name of the item being controlled by the slider.
- **SetAkSliderItemProperty**: Sets the property to be controlled by the slider.
- **SetAkSliderItemId**: Sets the ID of the item to be controlled (GUID string, using the "Digits with Hyphens in Braces" GUID format).
- **SetAkSliderRangeMin**: Sets the slider's minimum value.
- **SetAkSliderRangeMax**: Sets the slider's maximum value.
- **GetAkSliderRangeMin**: Gets the slider's minimum value.
- **GetAkSliderRangeMax**: Gets the slider's maximum value.
Editor Integration

Event Drag-and-Drop

Events and Auxiliary Busses can be dragged directly from the Wwise Picker into the Unreal Content Browser to create corresponding AkAudioEvent and AkAuxBus objects:

1. In the Unreal Editor, navigate to the Event or Auxiliary Bus using the Wwise Picker.
2. In the Unreal Editor, select the target Package in the Unreal Content Browser.
3. Drag the Event or Auxiliary Bus into the Unreal Content Browser: a corresponding AkAudioEvent or AkAuxBus object is created.
The WAAPI Picker

The Wwise Unreal integration makes extensive use of the Wwise Authoring API in the WAAPI-enabled Wwise Picker.

**Note:** If you do not wish to enable WAAPI on your machine, the old Wwise Picker will still be available. If a WAAPI connection is unavailable, the WAAPI picker will be disabled. If a WAAPI connection is available, the old Wwise Picker will be disabled.

Please refer to **Wwise Authoring API (WAAPI)** for instructions on how to enable WAAPI on your machine.

The WAAPI picker is populated in real-time from the running Wwise Authoring application. All changes made in Wwise are immediately reflected in the WAAPI picker, whether the Wwise project is saved or not.

The WAAPI Picker allows to do much more than the Wwise Picker. Here is an overview of all of its features.
Sound Preview

The WAAPI picker allows to preview sounds directly from Wwise, without having to generate SoundBanks. To do so, select an Event or an Actor-
Mixer from your project, and press the space bar. Alternatively, you can right-click on an Event or Actor-Mixer and select **Play/Stop**. To stop a playing sound, press space bar again, or right-click and select **Stop All**.

**Renaming Wwise Objects**

You may rename elements directly in your Wwise project. To do so, select the element you wish to rename in the WAAPI picker window, and press F2. Alternatively, you may right-click on the element and select **Rename**. You may then rename the object and press Enter to commit the changes. The name change will then immediately be reflected in the Wwise Authoring application.

**Deleting Wwise Objects**

You may delete elements directly in your Wwise project. To do so, select the element you wish to rename in the WAAPI picker window, and press Delete on your keyboard. Alternatively, you may right-click on the element, and select "Delete". The deletion will then immediately be reflected in the Wwise Authoring application.

**Finding the associated WorkUnit on disk**

You may locate the associated WorkUnit to an object in a file explorer using the WAAPI picker by right-clicking the element, and selecting "Show in Folder".

**Finding the associated object in Wwise**

You may focus a Wwise object directly in Wwise. To do so, select the element you wish to focus in Wwise, and press Ctrl+Shift+1 on your keyboard. Alternatively, you may right-click on the element, and select "Find in the Project Explorer". The element will then be focused in the currently running Wwise Authoring application.
Radius Visualization

Some workflow enhancements in the Unreal Editor are enabled through SoundFrame when running the Wwise Authoring Application alongside the Unreal Editor on the same machine.

With SoundFrame enabled and the Wwise Authoring Application running, the radius of the relevant assets are drawn when an AkAmbientSound is selected. If changes are made in Wwise Authoring, the radius is updated in real time:

1. Select an AkAmbientSound in the Actor Browser.
2. In the Wwise Authoring Application, make changes to any object associated with the Event assigned to the AkAmbientSound.
3. The corresponding sphere in the Unreal Editor resizes at the same time.

If the Wwise Authoring Application is not running on your machine, it is still possible to see the attenuation sphere in Unreal. In your Wwise project settings, under the SoundBanks tab, make sure to select Generate Per Bank Metadata File, Generate JSON Metadata, and Max Attenuation. The next time the SoundBanks are generated in the Unreal Editor, the Max Attenuation information will be added to all UAkAudioEvents, and the attenuation spheres will be visible in your level viewport.
Editor Listener

Starting with Wwise 2017.2.0, one listener is now activated when in edit mode. This allows, for instance, to properly preview Level Sequences containing positioned audio. This listener's position is determined by the camera position of the focused viewport. Clicking between multiple viewports will make the editor listener "jump" from one position to another.

This listener is also used in the Animation Editor viewport. This allows all 3D-positioned sounds to properly be previewed while working on animations.

Note that when starting a Play in Editor (PIE) session, this listener is deactivated, and the traditional listener on the Camera is enabled. When ejecting from a PIE session to start a Simulation in Editor (SIE) session, both the Editor listener and the Camera listener are active at the same time. Switching back to PIE mode will disable the Editor listener again.
Blueprint Functions

Several Wwise-specific global functions are exposed to scripting; the following are available in the Audiokinetic category:
Get Ak Component

Obtains an AkComponent that is attached to and follows a specified component. Several methods are available directly on the AkComponent : see AkComponent Blueprint Functions.
Post Event At Location

Posts a Wwise Event at a specified location. This is a fire-and-forget sound, created on a temporary Wwise Game Object with no corresponding AkComponent.
Set RTPC Value

Sets the value of a Game Parameter, optionally targeting the root component of a specified actor.
Set State

Sets the active State for a given State Group.
Spawn Ak Component at Location

Creates a new AkComponent at the specified location. By default, the component will automatically be destroyed once the Event it is playing is done. Parameter Auto Post controls whether to post the event immediately upon the component's creation (Default value is false). Advanced parameter Auto Destroy controls whether to destroy the component when the first Event posted on this component is done (Default value is true).

This Blueprint node is especially useful if you wish to set a Switch on a fire-and-forget sound. This behavior can be accomplished by disabling Auto Post, setting a Switch on the spawned Ak Component, and then posting the Event, like so:
Stop All

Stops all currently playing sounds.
Set Multiple Positions

Sets multiple positions to a single game object. Setting multiple positions on a single game object is a way to simulate multiple emission sources while using the resources of only one voice. This can be used to simulate wall openings, area sounds, or multiple objects emitting the same sound in the same area. Note: Calling SetMultiplePositions() with only one position is the same as calling SetPosition(). Parameter GameObjectAkComponent is the AkComponent of the game object on which to set positions. Positions is an Array of transforms to apply. MultiPositionType is the Position type. For more information on the different position types, refer to: MultiPositionType.
Set Multiple Channel Emitter Positions

Sets multiple positions to a single game object, with flexible assignment of input channels. Parameter GameObjectAkComponent is the AkComponent of the game object on which to set positions. ChannelMasks is an Array of channel masks to apply for each position. Positions is an Array of transforms to apply. MultiPositionType is the Position type.
Other

Other functions are available in sub-categories:

- Actor Blueprint Functions
- AkAmbientSound Blueprint Functions
- AkComponent Blueprint Functions
- SoundBank Blueprint Functions
- Debug Blueprint Functions
- WAAPI Blueprint Functions
Wwise Unreal Integration » Using the Integration » Features » Blueprint Functions
**Actor Blueprint Functions**

These Blueprint functions target the root component of an Actor. If no *AkComponent* is attached to the Actor's root component, one is created.
Post Event

Posts a Wwise Event attached to and following the root component of the specified Actor.
Post Trigger

Posts a Trigger to Wwise, targeting the root component of a specified Actor.
Set Occlusion Refresh Interval

Sets the time interval at which the AkComponent attached to the root component performs occlusion calculations. Set to 0 to turn off occlusion on the component.
Set Switch

Sets the active Switch for a given Switch Group, targeting the root component of a specified Actor.
Set Output Bus Volume

Set the output bus volume (direct) to be used for the specified game object. The Bus Volume value is a number ranging from 0.0f to 1.0f.
Stop Actor

Stops all sounds on the given Actor.
Use Reverb Volumes

Sets whether the **AkComponent** attached to the root component is influenced by **AkReverbVolume** or not.
AkAmbientSound Blueprint Functions

Several Wwise-specific actions can be executed on AkAmbientSound actors; these are available in the Ambient Sound category.
Start All Ambient Sounds

Starts playback of all AkAmbientSound present in the map.
Start Ambient Sound

Starts playback of the AkAmbientSound.
Stop All Ambient Sounds

Stops playback of all AkAmbientSound present in the map.
Stop Ambient Sound

Stops playback of the \texttt{AkAmbientSound}.
AkComponent Blueprint Functions

Several Wwise-specific actions can be executed on AkComponent scene components; these are available in the Ak Component category.
Get Attenuation Radius

Returns this AkComponent's effective attenuation radius (ScalingFactor * MaxAttenuation).
Post Ak Event

Posts the specified AkAudioEvent in Wwise.
Post Associated Ak Event

Posts this AkComponent's internal AkAudioEvent in Wwise.
Post Trigger

Posts a trigger on the associated *AkComponent*. 
Set Attenuation Scaling Factor

Sets the attenuation scaling factor, which modifies the attenuation computations on the game object to simulate sounds with a larger or smaller area of effect.
Set Listeners

Sets the listeners on the AkComponent.
Set Occlusion Refresh Interval

Sets the time interval at which the AkComponent performs occlusion calculations. Set to 0 to turn off occlusion on the component.
Set Output Bus Volume

Set the output bus volume (direct) to be used for the specified game object. The Bus Volume value is a number ranging from 0.0f to 1.0f.
Set RTPC Value

Sets the value of a Game Parameter for the associated AkComponent.
Set Stop when Owner Destroyed

Sets the StopWhenOwnerDestroyed value on the corresponding AkComponent.
Set Switch

Sets a Switch Group to a given Switch on the associated AkComponent.
Stop

Stops the `AkAudioEvent` associated with the `AkComponent` from playing.
Use Early Reflections

Enables (or disables) early reflection for this Ak Component.
Use Reverb Volumes

Sets whether the AkComponent is influenced by AkReverbVolume or not.
**SoundBank Blueprint Functions**

Several Wwise-specific actions can be executed on *AkAudioBank* scene components; these are available in the SoundBanks category.
Clear Banks

Results in all currently loaded AkAudioBank being unloaded synchronously, including the init SoundBank.
Load Bank

Loads a specified SoundBank.
Load Banks

Loads a specified list of SoundBanks. If SynchronizeSoundBanks is set to true, only the SoundBanks in Banks will remain loaded in the SoundEngine. This means that any SoundBank that was previously loaded, and that is not found in the Banks array, will be unloaded.
**Load Init Bank**

 Loads the Init SoundBank.
Unload Bank

Unloads a specified SoundBank.
Wwise Unreal Integration » Using the Integration » Features » Blueprint Functions
Debug Blueprint Functions

These Blueprint functions are used to help debug your game.
**Start Output Capture**

Starts a Wwise output capture. The output file will be located in the same folder as the SoundBanks.
Add Output Capture Marker

Adds a text marker in the output capture file.
Stop Output Capture

Stops a Wwise output capture. The output file will be located in the same folder as the SoundBanks.
Start Profiler Capture

Starts a Wwise Profiler capture. The output file will be located in the same folder as the SoundBanks.
Stop Profiler Capture

Stops a Wwise Profiler capture. The output file will be located in the same folder as the SoundBanks.
Wwise Unreal Integration » Using the Integration » Features » Blueprint Functions
**WAAPI Blueprint Functions**

These Blueprint functions are used to connect to the Wwise Authoring application using the Wwise Authoring API.

For more information on WAAPI and its features, refer to [https://www.audiokinetic.com/library/edge/?source=SDK&id=waapi.html](https://www.audiokinetic.com/library/edge/?source=SDK&id=waapi.html).
JSON objects

Since WAAPI makes extensive use of JSON, a Blueprint wrapper was written for the Unreal FJsonObject. For more information related to nodes found under "WAAPI Json Manager", please refer to the Unreal Engine documentation on FJsonObject.
Call Waapi

Call WAAPI and get/set information/parameters according to the chosen URI.

Parameters:

- **WaapiUri**: The function that will be called when an event that we would be aware of occurs.
- **WaapiArgs**: The arguments referenced by the `in_uri` function.
- **WaapiOptions**: Optional flag to get more information about the event that occurred.

**Return Value**: A JSON object that contains useful information about the call process to a specific Event. It returns error info if the call fails.
Register WAAPI Connection Lost Callback

Subscribe to WAAPI connection lost event. This event will be broadcast when the WAAPI connection is lost. This should be used to clean up any resources that use Waapi.

Parameters:

- **Callback**: The event to call when the connection is lost.
Register WAAPI Project Loaded Callback

Subscribe to WAAPI project loaded event. This event will be broadcast whenever the correct Wwise project is loaded (as defined by the Wwise Project Path in the Wwise Plugin Settings). This should be used to initialize any resources that use Waapi.

Parameters:

- **Callback**: The event to call when a connection is established.
Subscribe To Waapi

Allows clients to subscribe to notifications according to the event.

Parameters:

- **WaapiUri**: The reference to the event that we would be aware of when it occurs.
- **WaapiOptions**: Optional flag to get more information about the event that occurred.
- **Callback**: A delegate to be executed during the subscription event.

Return Values:

- **SubscriptionId**: The ID of this subscription.
- **Result**: A boolean to ensure that the subscription was successfully done.
- **ReturnValue**: A JSON object that contains useful information about the subscription process to a specific event. It returns error info if the subscription failed.
Unsubscribe

Unsubscribe from an event.

Parameters:

- **SubscriptionId**: ID of the current subscription to the event from which we want to be unsubscribed.

Return Values:

- **UnsubscriptionDone**: A boolean to ensure that the unsubscription was successfully done.
- **Return Value**: A JSON object that contains useful informations about the unsubscription process from a specific event, gets an error infos in case the unsubscription failed.

Generated by doxygen 1.6.3
Animation

An Animation Notify is used to trigger Wwise Events in animation sequences.

The Blueprint for the Animation Notify is located here:

.../Plugins/Wwise/Content/AnimNotify_AkEvent.uasset.

- To add an Animation Notify:
  1. In the Unreal Content Browser, open an animation sequence asset.
  2. Right-click the **Notifies** track in the animation sequence editor and select **Add notify > AkEvent** or **AkEventByName**.

- The animation notification has the following properties:
  - **Event** or **Event Name**: The AkAudioEvent to be posted.
  - **Attach Name**: Name of the socket that emits the AkAudioEvent; if not defined, the AkAudioEvent is emitted from the animation owner.
  - **Follow**: Whether the Event should either follow the mesh or be posted at a specific location.
Wwise Unreal Integration » Using the Integration » Features
Level Sequencer

The Sequence Editor within Unreal Engine 4 is a cinematic editing tool similar to Matinee. It allows users to add Tracks that can modify certain properties of Actors in a level.

For instructions on how to:

- create a Level Sequence, refer to https://docs.unrealengine.com/latest/INT/Engine/Sequencer/Overview
- add Tracks to a Level Sequence, refer to https://docs.unrealengine.com/latest/INT/Engine/Sequencer/Overview
Wwise Level Sequencer Tracks

The Wwise Unreal Engine 4 Integration adds two Tracks: AkAudioRTPC and AkAudioEvent. The AkAudioRTPC Track is used to set Game Parameter values, and the AkAudioEvent Track is used to post Wwise Events.

Both of these Tracks can be added as Master Tracks or associated to an Actor. When associated with an Actor, they perform their Wwise related functions on the UAkComponent attached to this Actor. When created as a Master Track, the AkAudioRTPC Track sets global RTPC values, and the AkAudioEvent Track posts Events on a "dummy" game object.

For AkAudioRTPC Tracks, Game Parameter curves can be modified using the built-in curve editor. Key frames can be added by placing the cursor at the appropriate location and pressing the Add New Key button (found on the right side of the listed Track).

An AkAudioRTPC track in the Curve Editor view

For AkAudioEvent Tracks, an AkAudioEvent section can be added by placing the cursor at the appropriate location and pressing the AkAudioEvent button (found on the right side of the listed Track).
Alternatively, **AkAudioEvent** assets can be dragged from the Content Browser directly onto an **AkAudioEvent** Track.
Sequencer WAAP-Enabled Features

The Unreal integration uses the Wwise Authoring API (WAAPi) to extend the Sequencer functionality. For more information on WAAPi and its use in the Unreal integration, refer to the [Wwise Authoring API (WAAPi)] page. When Unreal is connected to the Wwise Authoring tool via WAAPi, AkAudioEvent sections are able to display waveforms for Wwise events. When the Wwise Authoring application is not running, or the Unreal integration is not connected to WAAPi, the AkAudioEvent sections will only display the Event name.

AkAudioEvent Section Waveform Displays

AkAudioEvent sections display audio waveforms for Events that contain Audio Sources. The waveform in an AkAudioEvent section shows the audio data for the longest Audio Source that is contained in the Wwise Event. For example, consider a Wwise Event called "Play_Sound" with three Audio Sources, "Layer_1", "Layer_2", and "Layer_3".

![Example Wwise Event with three Audio Sources](image)

If an AkAudioEvent section is added that has its Event property set to "Play_Sound", then the longest of the three Audio Sources will be displayed within the section. Note that the waveform displayed in the section may be followed by empty space if the Wwise Event lasts longer than the longest Audio Source that it contains. In this example, since the "Play_Sound" Event has two play actions (for "Layer_2" and "Layer_3") that are delayed, it will last longer than the "Layer_1" audio source that it contains. The empty space after the waveform indicates the maximum estimated duration of the Wwise event, as calculated during soundbank
generation (see **Wwise Project Setup**).

**Example AkAudioEvent Section in Sequencer**

If the length of the **AkAudioEvent** section is increased beyond the maximum estimated duration of the Wwise Event, the section will contain either a flat white line, or repeating diagonal lines. The flat white line indicates that the **Retrigger** property is disabled. The repeating diagonal lines indicate that the **Retrigger** property is enabled. The **Retrigger** property determines whether Wwise events are retriggered in the Sequencer, after they have finished (see **AkAudioEvent Section Properties**).

**Two AkAudioEvent Sections in Sequencer. The 'Retrigger' section has Retrigger enabled. The 'No_Retrigger' section has Retrigger disabled.**

"Out of Sync" Waveforms

If changes have been made to an Event's workunit in the Wwise project that are not reflected in the generated soundbanks, any **AkAudioEvent** sections for that event will be marked as "out of sync". The waveform will be displayed in red, and an asterisk will be appended to the name of the AkAudioEvent section. There will also be a warning printed to the output log when Events are triggered from "out of sync" sections.

**Example 'Out of Sync' AkAudioEvent Section**
In order to keep AkAudioEvent sections in sync with the Wwise project, there are options to save the project and regenerate soundbanks, directly from the Sequencer. In order to synchronise an AkAudioEvent section, open the section's context menu (right click) and select "Save Wwise project and refresh" (see AkAudioEvent Section Properties). In order to synchronise all of the sections in an AkAudioEvent track, open the track's context menu (right click) and select "Save Wwise project and refresh all sections" (see AkAudioEvent Track Context Menu Options). Note that for these options to be available, the AkAudioEvent asset must have its Required Soundbank set to a valid Audiokinetic Bank asset. For more information on AkAudioEvent assets and their Required Bank property, refer to the AkAudioEvent page.
**Scrubbing**

The Sequencer integration supports scrubbing forwards and backwards over AkAudioEvent tracks. By dragging the Sequencer playhead over AkAudioEvent sections, scrub snippets can be heard. The length of these scrub snippets can be varied using the **Scrub Tail Length Ms** property, accessible from the AkAudioEvent section's context menu (right click). Refer to **AkAudioEvent Section Properties** for further details.
AkAudioEvent Track Context Menu Options

From the AkAudioEvent track, a context menu can be opened (right-click) which includes the following Audiokinetic-specific options:

- **Save Wwise project and refresh all sections**: This will save the Wwise project in the Authoring application, via WAAPI, and then re-generate the required soundbanks for all AkAudioEvent sections in the sequencer track. The sections will then update their waveform displays.
AkAudioEvent Section Context Menu Options

From the AkAudioEvent section on the track, a context menu can be opened (right-click) which includes the following Audiokinetic-specific options:

- **Save Wwise project and refresh**: This will save the Wwise project in the Authoring application, via WAAPI, and then re-generate the required soundbank for this AkAudioEvent section. The waveform display will then be updated.
- **Match section length to Wwise event length**: This will match the length of the section in the Sequencer track to the duration of the Wwise Event.

AkAudioEvent Section Properties

The properties list in the context menu includes the following Wwise-specific properties for the section:

- **Ak Audio Event**: Lists the following editable properties of the AkAudioEvent for the selected section:
  - **Event**: Event icon: Hover over the Audiokinetic Event icon to get the path to the Event. Double-click the icon to open the Event in the editor.
  - **Event**: Audiokinetic Event: Open this list to create a new Audiokinetic Event, edit or copy the currently selected Event, or browse to another existing Audiokinetic Event.
  - **Retrigger Event**: When this box is checked, the Wwise event in the Section will be retriggered when the Sequence plays beyond the end of the event (off by default).
  - **Scrub Tail Length Ms**: This defines the duration in milliseconds of the scrub snippets that are played when scrubbing over the sequence (default = 100ms).
  - **Max Source Duration**: Displays the duration of the longest Audio Source contained in the Wwise Event (read only).
  - **Advanced**:
    - **Event Name**: Enter the exact name of the desired Event.
This field is ignored if an Event is already selected in the Audiokinetic Event list.

- **Movie Scene Ak Audio Event Section:**
  - **Stop at Section End:** Uncheck this box to keep the Event playing when the end of the section is reached (on by default).
Known Issues and Constraints

Generally it is recommended to use very simple AkAudioEvent sections in the Level Sequencer, such as a simple Play action on a Sound SFX. More complex Wwise Events, such as those containing delayed actions, Seek actions, or which reference Random Containers, Sequence Containers and other non-deterministic objects should be avoided or broken down into simpler events for optimal use in the Level Sequencer.

Scrubbing and playing from cursor will not work as expected on Events with delayed actions or which contain infinitely looping sounds.

Play In Editor Limitations

If a level sequence is played from the Sequencer editor window while Unreal is running in Play in Editor (PIE) mode, any AkAudioEvent tracks that are associated to game objects will not trigger events. In order to hear AkAudioEvent tracks that are bound to game objects, the level sequence must be triggered from the Game World.
Wwise Project Setup

To ensure that Event lengths are properly represented in the AkAudioEvent Tracks, the Wwise project needs to be set up to estimate the duration of its audio Events and JSON metadata must be generated. To enable these project-wide settings, go to the Project Settings’ SoundBanks tab. Enable the **Estimated duration** check box within the **Metadata Options** group box and enable the **Generate JSON Metadata** check box within the **SoundBank Settings** group box, as seen below.
Required Project Settings for AkAudioEvent Tracks
Matinee

Two Wwise-specific Track types are introduced to Matinee:
AkAudioEvent Track

The AkEvent Track triggers Wwise Events at particular points in the sequence. When placing a new key, the AkAudioEvent currently selected in the Unreal Content Browser is used.
AkRTPC Track

The AkRTPC Track lets you control a Wwise Game Parameter using a curve in the sequence.

- Properties
  - **Param**: The name of the Game Parameter to control. This name should exactly match the name of a Game Parameter in the Wwise Project.
Occlusion

Basic occlusion is exposed in UAkComponent::SetOcclusion(). To enable occlusion on an Actor, use the Set Occlusion Refresh Interval function in the Blueprint Editor. A version for a UAkComponent is also available. If the refresh interval is set to 0, the occlusion check will never be performed.

Determining whether a listener is occluded from a source is done with a simple line-of-sight check, using the line trace channel set in the AkComponent's properties (OcclusionCollisionChannel). If the line of sight is blocked, the occlusion level calculation is started. This calculation maps the hit point on the obstacle to its bounding box, creating twelve points around the obstacle. To see if these "secondary paths" are also blocked, other line-of-sight tests are done. The occlusion sent to the SoundEngine is modulated from the number of "secondary paths" that are blocked.

A temporal fade method is also added to allow smooth transition between occlusion levels. To change the fade speed, change the OCCLUSION_FADE_RATE constant.
The Wwise Authoring API is used to communicate with the Wwise authoring application. WAAPI allows the Unreal integration to request and alter information about the Wwise project during a session, provided that both Unreal and Wwise Authoring are running, and Unreal is connected to WAAPI.

For more information on WAAPI and its features, refer to https://www.audiokinetic.com/library/edge/?source=SDK&id=waapi.html.

**Note:** WAAPI is available only on the Windows and Mac platforms, both in Editor and in Game.
Connecting to WAAPi

The Audiokinetic Unreal integration is set to automatically connect to WAAPi. In order to disable this functionality, the "Auto Connect to WAAPi" check box should be unchecked in the **Wwise Plugin Settings** menu. This menu is accessible from the Unreal project settings menu (Edit->Project Settings...). In the left panel, scroll down to the Plugins section and click on Wwise.

For WAAPi-enabled features to run, the Wwise project opened in the Wwise Authoring application must match the project defined by the **Wwise Project Path** in the **Wwise Plugin Settings**.
Using WAAPI from C++

An Unreal wrapper class for the WAAPI C++ sample client is provided as part of the AkAudio module.

In order to be able to use it, you first must add the AkAudio module as a dependency to your game, by following the steps outlined in UE4 C++ projects. Once this is done, the FAkWaapiClient class will be available.
Frequently Asked Questions

Q: Where are banks generated when I generate them from the Unreal editor or when cooking?
A: By default, they are generated in the following location:

```
UE4/[GameName]/Content/WwiseAudio/[Platform]
```

Some banks containing language assets may be in language-specific folders.

Q: I am using the new custom platforms feature in Wwise 2015.1, and my SoundBanks are not loading. Why?
A: With the custom platforms feature, the SoundBanks are generated in a sub-folder with the same name as your custom platform. You will need to modify the integration code to point to that folder. Open the file .../Wwise/Source/AkAudio/Private/AkAudioDevice.cpp, and search for the FAKAudioDevice::SetBankDirectory method. In this method, you will see the various default sub-folders for the supported platforms. Modify the one corresponding to your custom platform, and recompile the integration.

Q: When I generate banks from the Wwise application, I noticed that fewer banks are generated than when generating banks from the Unreal Editor. Why?
A: When importing a SoundBank definition file, Unreal may ask Wwise to generate banks that do not exist in the Wwise project. If you want the Unreal-generated bank to persist in the Wwise project, you can add the command-line argument -Save when importing the SoundBank definition file.

Q: When I try to use a plug-in in Unreal Engine 4, I get the following error: LogAkAudio:Error: Plug-in not registered: 4163. How do I use plug-ins?
A: In order for your plug-in to register itself with the SoundEngine, you need to include the plug-in's factory header (.h) file. See Initializing the SoundEngine to learn about this.
Q: Can I debug my code using the Wwise SoundEngine debug libraries?
A: By default, the Unreal Engine does not use the debug C Runtime Libraries, but Wwise uses them. When building in debug, this would cause linking errors. This is why, by default, the Wwise Unreal integration uses the profile libraries. If you wish to link against the debug Wwise libraries, simply set the `bDebugBuildsActuallyUseDebugCRT` variable to `true` in the following file:

```
<UE4_ROOT>/Engine/Source/Programs/UnrealBuildTool/Configuration/BuildConfiguration.cs
```

Q: I can't connect to my game running on an Xbox One, and I get the following error in the log: "Could not find Wwise network ports in AppxManifest. Network communication will not be available." How can I connect the Wwise Authoring Application to my game?
A: You should make sure to add the required Wwise network ports to your game's AppxManifest.xml. To do so, ensure the following Xbox One platform settings are filled in your project settings (Edit > Project Settings):
<table>
<thead>
<tr>
<th>Secure Socket Descriptions</th>
<th>Array elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>WwiseDiscoverySocket</td>
</tr>
<tr>
<td>Available in Shipping Builds</td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td>Udp</td>
</tr>
<tr>
<td>Port range</td>
<td>24024</td>
</tr>
<tr>
<td>Can Initiate Connections</td>
<td></td>
</tr>
<tr>
<td>Can Accept Connections</td>
<td></td>
</tr>
<tr>
<td>Can Send Chat Data</td>
<td></td>
</tr>
<tr>
<td>Can Receive Chat Data</td>
<td></td>
</tr>
<tr>
<td>Can Send Game Data</td>
<td></td>
</tr>
<tr>
<td>Can Receive Game Data</td>
<td></td>
</tr>
<tr>
<td>Can Send Debug Data</td>
<td></td>
</tr>
<tr>
<td>Can Receive Debug Data</td>
<td></td>
</tr>
<tr>
<td>Can send insecure data (warning: requires certification)</td>
<td></td>
</tr>
<tr>
<td>Can send insecure data (warning: requires certification)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Array elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Available in Shipping Builds</td>
</tr>
<tr>
<td>Protocol</td>
</tr>
<tr>
<td>Port range</td>
</tr>
<tr>
<td>Can Initiate Connections</td>
</tr>
<tr>
<td>Can Accept Connections</td>
</tr>
<tr>
<td>Can Send Chat Data</td>
</tr>
<tr>
<td>Can Receive Chat Data</td>
</tr>
<tr>
<td>Can Send Game Data</td>
</tr>
<tr>
<td>Can Receive Game Data</td>
</tr>
<tr>
<td>Can Send Debug Data</td>
</tr>
<tr>
<td>Can Receive Debug Data</td>
</tr>
<tr>
<td>Can send insecure data (warning: requires certification)</td>
</tr>
<tr>
<td>Can send insecure data (warning: requires certification)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Array elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Available in Shipping Builds</td>
</tr>
<tr>
<td>Protocol</td>
</tr>
<tr>
<td>Port range</td>
</tr>
<tr>
<td>Can Initiate Connections</td>
</tr>
<tr>
<td>Can Accept Connections</td>
</tr>
<tr>
<td>Can Send Chat Data</td>
</tr>
<tr>
<td>Can Receive Chat Data</td>
</tr>
<tr>
<td>Can Send Game Data</td>
</tr>
<tr>
<td>Can Receive Game Data</td>
</tr>
<tr>
<td>Can Send Debug Data</td>
</tr>
<tr>
<td>Can Receive Debug Data</td>
</tr>
<tr>
<td>Can send insecure data (warning: requires certification)</td>
</tr>
<tr>
<td>Can send insecure data (warning: requires certification)</td>
</tr>
</tbody>
</table>
Use these Xbox One platform settings for the Secure Socket Descriptions
Use these Xbox One platform settings for the Secure Device Associations

As a result, your `<GAME>/Config/XboxOne/XboxOneEngine.ini` should contain the following lines:

```ini
+SecureSocketDescriptions=(SecureSocketName="WwiseDiscoverySocket"
+SecureSocketDescriptions=(SecureSocketName="WwiseCommandSocket"
+SecureSocketDescriptions=(SecureSocketName="WwiseNotificationSocket"
+SecureDeviceAssociations=(SecureDeviceAssociationName=
+SecureDeviceAssociations=(SecureDeviceAssociationName=
+SecureDeviceAssociations=(SecureDeviceAssociationName=
```

For more information on how to modify the Xbox One AppxManifest, please refer to the Unreal Engine documentation.

**Q:** My game freezes when I deploy it to an iOS, XboxOne, or Switch device. Why?

**A:** The Unreal Engine built-in audio system needs to be disabled. To do so, please follow the appropriate instructions for your platform:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>In <code>&lt;UE4_ROOT&gt;/Engine/Config/iOS/IOSEngine.ini</code>, replace \</td>
</tr>
<tr>
<td></td>
<td>AudioDeviceModuleName=IOSAudio with \</td>
</tr>
<tr>
<td></td>
<td>AudioDeviceModuleName=</td>
</tr>
<tr>
<td>XboxOne</td>
<td>In <code>&lt;UE4_ROOT&gt;/Engine/Config/XboxOne/XboxOneEngine.ini</code>, replace AudioDeviceModuleName=XAudio2 with AudioDeviceModuleName=</td>
</tr>
<tr>
<td>Switch</td>
<td>In <code>&lt;UE4_ROOT&gt;/Engine/Config/switch/BaseSwitchEngine.ini</code>, replace</td>
</tr>
</tbody>
</table>
Q: When I try to use the Wwise Convolution Reverb plug-in in Unreal Engine 4, why do I get the following error: LogAkAudio:Error: Plug-in media unavailable: MediaID?
A: For Unreal Engine 4.10 and earlier, plug-ins that require media (like the Impulse Response for Wwise Convolution Reverb) require that the SoundBank be edited in the Wwise application as documented on the AudioKinetic website. In Unreal Engine 4.11 and later, you can assign a SoundBank an Auxiliary Bus via the **Required Bank** field in the **UAkAuxBus** properties. This will add necessary plug-in media to the SoundBank.

Q: When running a packaged game, some or all of my sound are not playing. Why?
A: The Wwise plug-in generates the SoundBanks in the WwiseAudio folder in your game’s content folder. Since .bnk files are not recognized by Unreal, the WwiseAudio folder is added as an "Additional non-asset directory to package" in your game's packaging settings. For this reason, Unreal .uasset files (including AkAudioEvent and AkAudioBank) should **not** be placed in the WwiseAudio folder, as they will be incorrectly packaged twice.
Wwise Unreal Integration
Using Wwise Spatial Audio in Unreal

This tutorial presents new functionalities introduced by Wwise 2017.1 and the new Spatial Audio features introduced in the Unreal plug-in. It explains the workflow to integrate spatial audio components in a game, and provides technical information about initializing spatial audio features (such as 3D busses, 3D positioned Sound SFX, or the Reflect Effect plug-in) in the Wwise Authoring Tool and importing the Spatial Audio component in a game using the Unreal Engine.

It is assumed that the reader has a basic working knowledge of Wwise and the Unreal Engine. However, the content of this document is detailed enough for users of any level of expertise to complete this tutorial. Further information can be obtained from the Unreal Documentation.

Throughout this tutorial, four different virtual acoustic zones will be created, each of these virtual acoustic zones demonstrates one capability of the new Spatial Audio plug-in. Each area has an ambient sound placed at its center that acts as an emitter. The listener is left as the default and corresponds in this case to the camera, which can be moved around to explore the different acoustic environments.

Sections A, B, and C are required steps to build a proper working environment and are expected to be completed in this specific order. Any of the following sections (namely D, E, F, and G) can be completed separately and independently in any order, once the three first sections are completed.

Note: Completion of sections using the Reflect plug-in require the appropriate license.

- A - Setting Up The Project
- B - Preparation
- C - Add Reference Sound in Unreal
- D - Add Late Reverb
- E - Add Surface Reflector
- F - Add Rooms and Portals
- G - Add Spot Reflector
A - Setting Up The Project

Using the latest version of Wwise and Unreal, follow the steps below in order to build your working environment.

1. Launch Unreal from the Epic launcher
2. Create a new Unreal Project C++ Blank project (without starter content) called TestSpatialAudio.
3. Close Unreal.
4. Start the Wwise Launcher.
5. Install Wwise.
6. Select the Unreal Engine tab.
7. Click the **Integrate Wwise into Project...** button.
8. Launch Wwise using the **Open in Wwise** button.
9. Launch Unreal using the **Open in Unreal** button.
B - Preparation

In this section, we are scaling the floor and setting up walls and portals to delimit four different areas that will correspond to the different acoustic environments in which we will put the spatial audio components.

1. Select the floor and set a new Transform.
   - Set Location to (x=0.0, y=0.0, z=0.0).
   - Set Scale to (x=2.0, y=2.0, z=1.0).

2. Create walls using 8 cubes with the following dimensions and locations:
   - Location (x=775.0, y=0.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0).
   - Location (x=225.0, y=0.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0).
   - Location (x=-225.0, y=0.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0).
   - Location (x=-775.0, y=0.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0).
   - Location (x=0.0, y=-775.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0) Rotation (x=0.0, y=0.0, z=90.0).
   - Location (x=0.0, y=-225.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0) Rotation (x=0.0, y=0.0, z=90.0).
   - Location (x=0.0, y=225.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0) Rotation (x=0.0, y=0.0, z=90.0).
   - Location (x=0.0, y=775.0, z=100.0) Scale (x=4.5, y=0.25, z=2.0) Rotation (x=0.0, y=0.0, z=90.0).

3. Move Player start to:
   - Location (x=-500.0, y=-500.0, z=100.0).

4. In order to help differentiate the rooms, add the following PointLights at the specified locations with Intensity set to 50,000:
   - Red, Location (x=-500.0, y=-500.0, z=500.0).
   - Blue, Location (x=500.0, y=-500.0, z=500.0).
   - Green, Location (x=500.0, y=500.0, z=500.0).
   - Yellow, Location (x=-500.0, y=500.0, z=500.0).

5. Save as "NewMap".
Build working environment
C - Add Reference Sound in Unreal

The first zone introduces a 3D positioned sound which should be panned and occluded/obstructed accordingly as the listener is moving around the source. Occlusion/obstruction occurs when the listener (namely, the main camera) moves outside the area and hides behind opaque components such as a wall, a floor, or a ceiling.

1. In the Wwise project, create a new Sound SFX in the Default Work Unit of the Actor-Mixer Hierarchy, name it ReferenceSound, and import an impulse sound.

   ![Create Sound SFX]

2. Make sure to enable **Use game-defined auxiliary sends** with positioning enabled and type set to 3D.
3. Create the associated Event "PlayReferenceSoundTest", and drag and drop the "ReferenceSound" voice into it.

4. Save the project.

5. In Unreal, open Window > Wwise Picker and drag the
"PlayReferenceSoundTest" Event into the Content Browser.

Drag from Wwise Picker to the Content Browser

6. Double click the "PlayReferenceSoundTest" and create a new SoundBank called "TestSpatialAudioBank" directly from the SoundBank list.
Select SoundBank

7. Drag the "PlayReferenceSoundTest" Event into the scene to create a new AkAmbientSound.
   ○ Set Location to (x=-500.0, y=-500.0, z=100.0).

Create AkAmbientSound

8. Open Level Blueprint from the Blueprints menu and remove "Event BeginPlay" and "Event Tick".
9. Drag the newly created "PlayReferenceSoundTest" item from the
World Outliner into the blueprint.

10. Right-click the blueprint background and search for "Left Mouse Button".

11. Right-click the blueprint background again and search for "Post Event".

12. Connect the Pressed outlet to the Post Event Exec and the "PlayReferenceSoundTest" component to the Actor inlet.


Add user input to trigger ambient sound

14. In the upper menu, select **Build > Generate SoundBanks....** to generate the SoundBanks for Windows.
Generate SoundBanks from build menu

15. Make sure the banks are successfully generated in the Output Log.

Output Log for SoundBank generation

16. Start the scene. You should now hear the sound play, spatialized in 3D, when pressing the Left Mouse Button.
D - Add Late Reverb

The second area is a demonstration of using an AkSpatialAudioVolume to include a Late Reverb applied to an AkAmbientSound inside a contained area.

1. In the Wwise project, create an Auxiliary Bus called "TestReverbAuxBus" under the Master Audio Bus.

   ![Create an Auxiliary Bus for reverb](image)

2. Add a Wwise RoomVerb Effect and use the Cathedrals > Holy predefined ShareSet.
3. Create a second Event called "PlayLateReverbTest" and drag the "ReferenceSound" Sound SFX on it.
4. Save the project.
5. In Unreal, drag an AkSpatialAudioVolume volume in the scene from the Volumes selection.
   - Set Location to (x=500.0, y=-500.0, z=100.0).
   - Set Scale to (x=5.0, y=5.0, z=1.0).

6. Make sure only **Enable Late Reverb** is enabled and that **Enable
**Surface Reflectors** and **Enable Room** remain disabled for this AkSpatialAudioVolume.

![Image of Wwise settings: Surface Reflectors and Enable Room are disabled.](image)

Select 'Enable Late Reverb'

7. Drag the new TestReverbAuxBus and PlayLateReverbTest from the Wwise Picker into the Content Browser.

![Image of Content Browser with Drag and Drop action](image)

From Wwise Picker, drag Auxiliary Bus and Event

8. Double-click the PlayLateReverbTest and assign it to
Adding bus and Event to bank

9. Select the new AkSpatialAudioVolume and set the aux bus to TestReverbAuxBus in the Late Reverb section
Select late reverb aux bus

10. Drag the "PlayLateReverbTest" Event into the scene to create an AkAmbientSound.
   ▪ Set Location to (x=500.0, y=-500.0, z=100.0).

Add Event into the scene

11. Reproduce steps C-11 to C-13 for PlayLateReverbTest to trigger the
sound. (You can map a keyboard key instead of a mouse button.)
12. Regenerate the SoundBanks.
13. Start the scene and go inside the volume. Triggering
    PlayReferenceSoundTest should result in a sound with reverb
    applied on it.

    **Note:**
    If **Enable Room** is checked for the `AkSpatialAudioVolume`, the
    Auxiliary Bus containing the reverb must have 3D positioning
    enabled as well.
E - Add Surface Reflector

The third area demonstrates surface reflectors that are used to simulate the early reflection implied by the propagation of sound in an acoustic environment. The wave propagation is modeled using the image-source method which derives from the game geometry the appropriate time delay (distance), amplitude, and filter parameters in order to recreate a realistic acoustic environment. Real time computation evaluates the different paths acoustic waves take to reach the relative position of listeners in reference to sound emitters. The resulting effect provides a strong sensation of space of the acoustic environment. One can actually see the reflection through the Wwise Reflect Effect Editor, which displays a distance/amplitude representation of each reflection.

1. In Wwise, create a third Event called "PlayReflectTest" and drag the "ReferenceSound" Sound SFX on it.

2. Create an aux bus Called "TestReflectAuxBus" on the Master Audio Bus and put a Reflect plug-in on it.
Create new Event in Wwise

3. Make sure to select **Use game-defined auxiliary sends** and enable 2D positioning for this bus.

Enable 2D positioning for Auxiliary Bus

4. Set Reflect's Max Distance to 10,000 and Speed of Sound to 34,500.
Set Reflect Speed of Sound and Max Distance

5. Save Project
6. In Unreal, drag and drop the PlayReflectTest and TestReflectAuxBus from the Wwise Picker into the Content Browser
7. Double-click the PlayReflectTest Event and assign it to the TestSpatialAudioBank.
8. Drag and Drop the PlayReflectTest into the scene to create an AkAmbientSound:
   - Set Location to (x=500.0, y=500.0, z=100.0).
9. Drag n drop a AkSpatialAudioVolume into the scene
   - Set Location to (x=500.0, y=500.0, z=100.0).
   - Set relative scale to (x=5.0, y=5.0, z=1.0).
10. Make sure that "Enable Surface Reflectors" is enabled, and that "Enable Late Reverb" and "Enable Room" remain disabled for this AkSpatialAudioVolume.
Add spatial audio volume to scene

11. The PlayReflectTest AkAmbientSound should be inside the AkSpatialAudioVolume.
12. Select the PlayReflectTest AkAmbientSound, go to the Spatial Audio option and set "Early Reflection Aux bus" to "TestReflectAuxBus".
13. You can also enable Debug Draw and display 1st order reflections for this AkAmbientSound (in Ak Component > Spatial Audio > Debug Draw).
Debug draw option for reflections

14. Reproduce steps **C-11** to **C-15** for PlayReflectTest to trigger the sound. (You can map a keyboard key, instead of a mouse button.)
User input to trigger the sound

15. Regenerate the SoundBanks.
16. Start the scene and go inside the volume. You should see reflections being drawn and your sound being routed to an aux bus that uses the Reflect plug-in when triggering \texttt{PlayReflectTest}. 
F - Add Rooms and Portals

In a realistic acoustic environment, portals (such as windows or any other opening) will act as new sources for any listener outside the acoustic volume in which a sound is emitted. AkAcousticPortals are useful to model this situation using an Auxiliary Bus to route the audible content of a volume and expose it as a new 3 dimensional source in an adjacent acoustic space.

1. In the Wwise project, create a new Auxiliary Bus called "TestPortalAuxBus".
2. Make sure to select **Use game-defined auxiliary sends** and to enable 3D positioning.

![Create new Auxiliary Bus with 3D positioning](image)

3. Add a Wwise RoomVerb Effect with Room_Medium parameters.
Add reverb to Auxiliary Bus

4. Create a fourth Event called "PlayPortalTest" and drag the "ReferenceSound" Sound SFX on it.

Add new Play Event

5. Save the project.
6. In Unreal, drag and drop the PlayPortalTest from the Wwise Picker
7. Drag and drop the TestPortalAuxBus into the Content Browser.

8. Double click the "PlayPortalTest" Event and assign it to the TestSpatialAudioBank.

   - Set Location to (x=-500.0, y=500.0, z=100.0).
   - Set Absolute Scale to (x=5.0, y=5.0, z=1.0).

10. Select the "TestPortalAuxBus" as an aux bus in the Late Reverb.
Add the spatial audio volume to scene and select late reverb auxiliary bus

11. Make sure the **Enable Late Reverb** and **Enable Room** are both enabled and **Enable Surface Reflectors** remains disabled for this AkSpatialAudioVolume.
Make sure the room is enabled

12. Reproduce steps **F-1** to **F-7** to add another late reverb aux bus (with any RoomVerb preset) called "TestReferenceAuxBus".
Create new Auxiliary Bus with 3D positioning

Add reverb to Auxiliary Bus

13. Add another AkSpatialAudioVolume.
   - Set Location to (x=-500.0, y=-500.0, z=100.0).
   - Set Absolute Scale to (x=5.0, y=5.0, z=1.0).

14. Make sure **Enable Late Reverb** and **Enable Room** are both enabled and **Enable Surface Reflectors** remains disabled for this AkSpatialAudioVolume.
Add a second audio volume and enable room

15. Select the "TestReferenceAuxBus" as an aux bus in the Late Reverb section.

Select late reverb auxiliary bus

   - Location (x=-500.0, y=0.0, z=100.0) Scale (x=0.5, y=0.5, z=1.0).
   - Location (x=0.0, y=-500.0, z=100.0) Scale (x=0.5, y=0.5, z=1.0) Rotation (x=0.0, y=0.0, z=90.0).
   - Location (x=-500.0, y=0.0, z=100.0) Scale (x=0.5, y=0.5, z=1.0) Rotation (x=0.0, y=0.0, z=90.0).
Drag a new acoustic portal into the scene

17. Select the new Portal and set the initial state to Open in the "Ak Acoustic Portal" section.

Make sure the portal is opened

18. Drag and drop the PlayPortalTest into the scene.
   ○ Set Location to (x=-500.0, y=500.0, z=100.0).

Drag new Event into the scene

19. Reproduce steps **C-11** to **C-13** for PlayPortalTest to trigger the sound. (You can map a keyboard key, instead of a mouse button.)
20. Regenerate the SoundBanks for Windows.
21. Start the scene and stay at the start location. You should hear the sound from the portal when triggering PlayPortalTest.

**Note:**
An AkAcousticPortal must be oriented in a way that the rooms it will link between are positioned on its local Y axis. When selecting a portal, a yellow ribbon appears around the portal to help visualize that. The yellow line represents the separation between the front and back areas. The highest priority room will be chosen in case of overlapping rooms.

**Note:**
If the world contains one or more spatial audio rooms, then the behavior of the occlusion/obstruction algorithm changes to utilize the additional information that spatial audio rooms provide. If the line-of-sight test between the emitter and listener fails:

1. When the listener and emitter are in the same room, the Wwise obstruction filter (dry path only) is set.
2. When the emitter and listener are in different rooms, the Wwise obstruction filter (both wet and dry path) is set.
3. In the absence of spatial audio rooms, the algorithm assumes that all sounds that do not have a line of sight to the listener are occluded, and the Wwise occlusion filter (both wet and dry path) is set.

**Note:**
Having added a room for section F, you will now notice that the sounds created in previous sections that lie outside the new rooms are occluded when the listener is inside the room. Similarly, the sounds inside the rooms are occluded while the listener is outside the room.

In Wwise, you can fine tune the filter response of portal shadow region under the Obstruction/Occlusion tab in Project Settings.
Obstruction volume curve
Obstruction LPF curve
Occlusion volume curve
Occlusion LPF curve
G - Add Spot Reflector

Spot reflectors are the last feature introduced in this tutorial and are mainly used to model singular acoustic reflections. They can be used to model distant virtual man-made or naturally created geographic features which create a singular distinctive acoustic echo. For example, mountains or large walls. They work best when the emitter is the listener. A spot reflector can simply be dropped inside the environment for the engine to compute the appropriate acoustic delay between the emitter and the listener, based on the physical distance between their respective location and the point reflection.

1. In the Wwise project, create a new Auxiliary Bus called "TestSpotReflectorAuxBus" and apply a Reflect plug-in to it.

2. Make sure to enable **Use game-defined auxiliary sends** and enable 2D positioning.
3. Set Reflect's Max Distance to 10,000 and Speed of Sound to 34,500.

4. Save the project.

5. In Unreal, from the Wwise Picker, drag and drop the TestSpotReflectorAuxBus into the Content Browser.

6. Open the Level Blueprint to:
Enable Spot Reflectors on the player emitter.
Post the "PlayReferenceSoundTest" Event from the player emitter. (You can map a keyboard key instead of a mouse button.)

Enable Spot Reflectors on the player component and Post PlayReferenceSoundTest from it

7. Drag a new AkSpotReflector object into the scene.
   - Set Location to (x=-150.0, y=-150.0, z=100.0).
8. On the AkSpotReflector Object, set the option Aux Bus to your "TestSpotReflectorAuxBus".

9. Regenerate the SoundBanks.
10. Start the scene. When triggering the new Event from the player emitter, you should hear reflections from the spot reflector when both the emitter and the spot reflector are in the same room. Opening the Reflect plug-in Effect Editor on the "TestSpotReflectorAuxBus" Auxiliary Bus, you should see the reflection appear in the list and the graph.
What's New?

Each version of this integration matches a specific build of Unreal Engine 4. Here is what has changed in each release of the integration (in addition to upgrading to the new Unreal build).

Note: This integration does not support experimental Unreal Engine 4 features.

Migration Notes

Known Issues

Versions:

- Unreal Engine 4.17/4.18 - Wwise 2017.2.1.6524.866
- Unreal Engine 4.17/4.18 - Wwise 2017.2.0.6500.836
- Unreal Engine 4.15/4.16/4.17/4.18 - Wwise 2017.1.4.6407.760
- Unreal Engine 4.15/4.16/4.17/4.18 - Wwise 2017.1.3.6377.732
- Unreal Engine 4.15/4.16/4.17 - Wwise 2017.1.3.6377.715
- Unreal Engine 4.15/4.16/4.17 - Wwise 2017.1.2.6361.696
- Unreal Engine 4.15/4.16/4.17 - Wwise 2017.1.1.6340.673
- Unreal Engine 4.15/4.16 - Wwise 2017.1.0.6302.628
- Unreal Engine 4.12/4.13/4.14/4.15/4.16 - Wwise 2016.2.4.6098.451
- Unreal Engine 4.12/4.13/4.14/4.15 - Wwise 2016.2.3.6077.422
- Unreal Engine 4.12/4.13/4.14/4.15 - Wwise 2016.2.2.6022.359
- Unreal Engine 4.12/4.13/4.14 - Wwise 2016.2.0.5972.301
- Unreal Engine 4.12/4.13 - Wwise 2016.2.0.5972.274
- Unreal Engine 4.11/4.12 - Wwise 2016.1.2
- Unreal Engine 4.11/4.12 - Wwise 2016.1.1
- Unreal Engine 4.11.2 - Wwise 2016.1
- Unreal Engine 4.11 - Wwise 2015.1.7
- Unreal Engine 4.11 - Wwise 2015.1.6
- Unreal Engine 4.10 - Wwise v2015.1.4
- Unreal Engine 4.9 - Wwise v2015.1.2
- Unreal Engine 4.8 - Wwise v2015.1
- Unreal Engine 4.8 - Wwise v2014.1.5
- Unreal Engine 4.7 - Wwise v2014.1.3
- Unreal Engine 4.6 - Wwise v2014.1.1
- Unreal Engine 4.5 - Wwise v2014.1
- August 2014 - Wwise v2014.1
- August 2014 - Wwise v2013.2.9
- July 2014 - Wwise v2013.2.9
- June 2014 - Wwise v2013.2.8
- April 2014 - Wwise v2013.2.7
- March 2014 - Wwise v2013.2.6
- January 2014 - Wwise v2013.2.5
- December 2013 - Wwise v2013.2.4
- October 2013 - Wwise v2013.2.1
- September 2013 - Wwise v2013.2.1
- August 2013 - Wwise v2013.2
- July 2013 - Wwise v2013.1.1
- June 2013 - Wwise v2013.1.1
- May 2013 - Wwise v2013.1.1
- March 2013 - Wwise v2013.1
Unreal Engine 4.17/4.18 - Wwise
2017.2.1.6524.866

- **WG-34960** Removed min and max properties in AkSlider widget, which are now automatically set via WAAPI.
- **WG-35238** Fixed: All open AkEvent Sequencer segments update to dirty when changes are detected in their Work Units.
- **WG-35773** Added bounds to the `UAkComponent::UseEarlyReflections order` parameter.
- **WG-35949** Added ability to generate a single SoundBank containing only an Auxiliary Bus.
- **WG-36083** Removed WAAPI log spam when generating SoundBanks while the Sequencer window is open.
- **WG-36200** Fixed crash when running editor with `-game` flag.
- **WG-36357** Changed Launcher to now copy Visual Studio 2017 dependencies to `ThirdParty` folder.
- **WG-36415** Fixed crash in Unreal when adding new AkSlider in UMG
Unreal Engine 4.17/4.18 - Wwise
2017.2.0.6500.836

- Improved listener handling while in the Editor. See Editor Listener for more information.
- Added a new WAAPI-enabled Wwise picker. See The WAAPI Picker for more information.
- Improved Sequencer integration with scrubbing and 'play from anywhere' support. See Level Sequencer for more information.
- Added WAAPI-enabled waveform rendering to AkAudioEventSection Sequencer sections. See Level Sequencer for more information.
- Exposed WAAPI to Blueprint. See WAAPI Blueprint Functions for more information.
- Added WAAPI-enabled widgets for use in UMG. See WAAPI Widgets for more information.
- **WG-30009** Fixed: Added support for multiple listeners via Blueprint.
- **WG-30010** SetMultiplePositions is now exposed in Blueprint.
- **WG-33932** Removed the experimental alternate occlusion feature.
- **WG-35307** Added more configurations to the Set Bus Config Blueprint node.
• **WG-33333** Removed obsolete code that tried to handle global focus.
• **WG-34745** Reduced Lower Engine memory pool size on mobile platforms.
• **WG-34879** Exposed the collision channel used for occlusion line trace in AkComponent's properties.
• **WG-35035** Ensured setting the occlusion refresh interval to 0 from Blueprint correctly disables the feature.
• **WG-35104** In a multiplayer scenario, listeners are now properly handled.
• **WG-35463** Fixed: Portals not working on 32-bit platforms.
• **WG-35473** Now register game objects only when the world is valid.
• **WG-35614** Removed coordinates conversion. See migration_to_2017_1_4 for more information.
Unreal Engine 4.15/4.16/4.17/4.18 - Wwise
2017.1.3.6377.732

- Added support for Unreal 4.18.
- **WG-35104** Fixed: Do not create a listener on a dedicated server.
- **WG-35272** Fixed: Added support for Android 64-bit builds.
- **WG-35286** Fixed: Avoid crash when having a Add Surface Reflector Set Component Blueprint node connected to nothing.
Unreal Engine 4.15/4.16/4.17 - Wwise
2017.1.3.6377.715

- **WG-34797** Undoing a surface deletion on a Spatial Audio Volume properly refreshes its details panel.
- **WG-34810** Late Reverb Component properties are now properly serialized.
- **WG-34878, WG-34906** Sounds are now properly spatialized in a Play in Editor session.
- **WG-34907** Performance improvement for FAKAudioDevice::Get().
Unreal Engine 4.15/4.16/4.17 - Wwise
2017.1.2.6361.696

- **WG-32413** Fixed: Cannot hear sounds in the Content Browser and the Animation Editor.
- **WG-33970** Fixed: (Mac) Crash when running game with PAK files.
- **WG-34030** Fixed: Properly update the details panel when a surface is removed from an AkSpatialAudioVolume.
- **WG-34083** Fixed: Refresh issue when changing a Spatial Audio Volume's geometry properties.
- **WG-34213** Fixed: Sounds can now be heard from the Content Browser and the Animation Editor.
- **WG-34222** Virtual Acoustics factory ShareSets now appear in the Wwise Picker.
- **WG-34276** Added ability to allow posted Events to continue playing past their associated sections within Level Sequences.
- **WG-34605** Fixed: Portals are not pushed to Wwise Spatial Audio when starting a game, in some circumstances.
- **WG-34630** Fixed: Crash when `-nosound` option is enabled.
- **WG-34703** Fixed: Prevent crash when modifying multiple SpatialAudioVolumes at the same time.
- **WG-34704** Fixed: Prevent crash when unregistering an AkComponent that is a default listener.
- **WG-34745** Fixed: Reduced lower engine memory pool size.
• Added support for Unreal Engine 4.17
• Since Unreal 4.17, the Unreal audio system needs to be disabled on the Xbox One and Switch platforms. Please refer to Frequently Asked Questions for more information.
• **WG-34098** (Mac) An AkComponent located at exactly the same position as a volume with an AkRoomComponent will be properly assigned to that room.
• **WG-34119** Fixed: Uninitialized listener ID value sent to SpatialAudio API via SetEmitterAuxSendValues.
• **WG-34368** Fixed: FAkAudioDevice::PostEventAtLocation does not register game object.
• **WG-34388** Fixed: Crash in Editor when resetting RTPC sequence name to default in Level Sequencer.
Unreal Engine 4.15/4.16 - Wwise
2017.1.0.6302.628

- Added Spatial Audio components. See Spatial Audio Objects for more information.
- **WG-32095**: Added support for Visual Studio 2017
- **WG-32388** Fixed: Properly parse Max attenuation radius when generating SoundBanks.
Unreal Engine 4.12/4.13/4.14/4.15/4.16 - Wwise 2016.2.4.6098.451

- Added support for Unreal Engine 4.16
- **WG-31942** Fixed: Added support for Wwise file packages.
- **WG-33251** Fixed: Added missing include files when building non-Unity builds.
2016.2.3.6077.435

- **WG-32914** Fixed: Avoid potential deadlocks and crashes in the AK Unreal IO code in a packaged game.
Unreal Engine 4.12/4.13/4.14/4.15 - Wwise 2016.2.3.6077.422

- **WG-30020** Fixed: Allow possibility to dismiss the warning about missing Wwise project on editor startup.
- **WG-30695** Added support for the Switch platform.
- **WG-31076** Fixed: Marked all of the Wwise Blueprint nodes as BlueprintCosmetic to avoid running them on a dedicated server.
- **WG-31455** Added AkComponentCallbackManager. This allows better handling of required callbacks for an AkComponent, reducing concurrency risks.
- **WG-32046** Fixed: Add possibility to automatically start an AkAmbientSound on BeginPlay.
- **WG-32490** Fixed: Deprecated level sequencer code.
- **WG-32763** Fixed: Discontinued use of monolithic engine header files, like "Engine.h".
- **WG-32768** Fixed: Crash when loading large banks using EDL.
- **WG-32799** Fixed: Increased number of concurrent IO transfers in editor to speed up SoundBank loading.
• **WG-32464** Allow opening an existing project in UE4.15 on the Mac platform.
2016.2.2.6022.359

- **WG-31087** Added Level Sequence support.
- **WG-31687** Added Event Driven Loader support.
- **WG-31816** Added migration of Matinee tracks to Sequencer tracks.
- **WG-31924** Fixed: Editor crash when trying to post an Event on an AkComponent that is being auto-destroyed.
- **WG-32259** Fixed: Made AkComponent a Blueprintable component. Blueprint components can now use AkComponent as a base class.
2016.2.1.5995.317

- **WG-31588** Fixed: Make sure no sound is played on a dying actor.
- **WG-31590** Improved low-memory conditions related to callbacks.
- **WG-31860** Fixed: Remove warnings found by clang static code analysis.
- **WG-31876** Fixed: Allow relative paths for the Wwise install path.
- **WG-31888** Fixed: Potential crash on exit due to accessing deleted FString.
Fixed:

- **WG-31589** Fixed: Do not destroy a spawned component that is not set to auto-destroy and is failing to post an event.
- **WG-31678** Fixed: Streaming files can now be opened correctly.
- **WG-31771** Fixed: Avoid a crash when adding an AnimNotify_AkEvent to a playing animation.
Unreal Engine 4.12/4.13 - Wwise
2016.2.0.5972.274

- **WG-29980** Language-specific folders are now parsed to retrieve max attenuation values for Events.
- **WG-30448** Improved logging of AkComponent when attempting to post an invalid AkEvent.
- **WG-30491** Fixed: Make sure to stop sounds when ending a PIE session.
- **WG-31030** Fixed: Optimized FAkAudioDevice::Get().
- **WG-31040** Fixed: Memory leak in AkComponent when attempting to post an invalid AkEvent.
- **WG-31075** Fixed: Removed call to FAkAudioDevice::Get() from bank load callbacks in order to prevent crash in module manager.
- **WG-31186** Fixed: Prevent crash in AkComponentCallback by canceling Event callbacks when AkComponent gets destroyed.
- **WG-31204** Fixed: Memory leak when a spawned AkComponent failed to post its associated Event.
- **WG-31277** Fixed: Crash in editor when attempting to post an Event on a destroyed actor.

- **WG-30993** Fixed: Remove explicit LoadLibrary on XAudio 2.7.
- **WG-31015** Fixed: Link with the Recorder plug-in.
Unreal Engine 4.11/4.12 - Wwise 2016.1.2

- **WG-30304** Fixed: "Unload stream level" no longer posts a global "Stop All" to the SoundEngine.
- **WG-30754** Fixed: FAkAudioDevice::PostEvent now always returns the playing ID.
- **WG-30804** Fixed: Removed dependencies on the Wwise SDK samples. The new IO system now uses Unreal IO utilities only.
Unreal Engine 4.11/4.12 - Wwise 2016.1.1

- **WG-29972** Fixed: Multiple potential threading issues with the auto-destroy behavior of the AkComponent.
- **WG-29979** Fixed: EndOfEvent callbacks are now always called.
- **WG-30004** Fixed: SetGameObjectOutputBusVolume is now exposed in Blueprints and AkAudioDevice.
- **WG-30404** Fixed: The attenuation scaling factor now properly works on AkComponents.
- **WG-30409** Fixed: It is now possible to decode Vorbis-encoded files.
Unreal Engine 4.12 - Wwise 2016.1

- **WG-29991** Added AkEvent String Input Field to AkAmbientSound.
- **WG-29997** Suppressed duplicate "LogAkAudio: StopAll API called" entries in the output log.
- **WG-30218** Fixed: Crash when connecting the Wwise Profiler to the Android platform.
- **WG-30255** Fixed: UE4 crash when adding key to Ak Event Track in Matinee.
- **WG-29991** Added AkEvent String Input Field to AkAmbientSound.
- **WG-29997** Suppressed duplicate "LogAkAudio: StopAll API called" entries in the output log.
Unreal Engine 4.11.2 - Wwise 2016.1

- **WG-29917** Fixed: Fixed a case where the attenuation radius of AkAmbientSounds did not show up in the Editor window.
- **WG-30000** Fixed: Changed meta properties of StartAllAmbientSounds and StopAllAmbientSounds.
- **WG-30012** Fixed: Integrated GitHub pull request #5. Show project supported platforms as "Available Platforms" in the "Generate Sound Banks" window.
- **WG-30014** Fixed: Fixed a crash when starting a Play in Editor session that uses a dedicated server.
- **WG-30031** Fixed: Removed usage of the World global pointer.
- **WG-30205** Fixed: On the Mac platform, the Wwise.app path may now contain spaces.
Unreal Engine 4.11 - Wwise 2015.1.7

- **UI-316** Fixed: Prevent a crash when starting a Play in Editor session with a dedicated server.
- **WG-29917** Fixed: Make sure the attenuation radius is visible in the Editor.
- **WG-29944** Fixed: Stop ambient sounds when stopping PIE session.
The Wwise Unreal integration is now a plug-in. Please see Migrating from the UE4 Wwise integration source code to the plug-in version for migration notes.

**UI-273** Fixed: Create a new uasset type for Auxiliary Busses. This allows to assign an AuxBus to a SoundBank. See AkAuxBus for more information.

**UI-280** Fixed: Deprecated "...by name" methods, and replaced then with a string field in the relevant methods. See Migrating "...by name" methods for more information.

**UI-309** Fixed: Made the Wwise project path relative to the game folder, instead of relative to the UE4Editor.exe file. See Migrating the Wwise project path for more information.
Unreal Engine 4.10 - Wwise v2015.1.4

- **UI-265** Fixed: In the AnimNotifies, test the AkComponent's validity before using it. This reduces log spam when using UE with the `-nosound` option.
- **UI-270** Fixed: It is now possible to build in non-Unity mode.
Unreal Engine 4.9 - Wwise v2015.1.2

- **UI-249** Fixed: Avoid crashing when no Audio Playback device is available on Windows.
- Added a "Spawn AkComponent at location" Blueprint node. See [Spawn Ak Component at Location](#) for more information.
- Added support for the Mac Editor.
- Added support for the Mac platform.
- Added support for the iOS platform.
- Added support for the Linux platform.
Unreal Engine 4.8 - Wwise v2015.1

- Migrated the UE4.8 integration to Wwise 2015.1
- Fixed a crash on Xbox One if the app manifest did not contain a definition for Wwise communication ports.
Unreal Engine 4.8 - Wwise v2014.1.5

- **UI-206** Fixed: Sounds with an attenuation can now be heard when an animation window is opened.
- **UI-212** Fixed: Added a "Load Init Bank" Blueprint node.
- **UI-213** Fixed: Added null checks in Blueprint nodes, preventing crashes.
- **UI-214** Fixed: No crash when saving an output capture to disk.
- **UI-215** Fixed: No issue with the Attenuation Scaling Factor of an AkComponent not being set when attached to an actor.
- **UI-217** Fixed: Made AkReverbVolume's collision settings visible in the details panel.
- **UI-220** Fixed: Fixed the occlusion fade behavior for sounds that a spawn occluded.
- **UI-223** Fixed: The gathering of the max attenuations for AkEvent does not halt when the parsing of one SoundBank fails.
- **UI-226** Fixed: SoundBank generation is now performed by the 32-bit WwiseCLI.exe when the 64-bit WwiseCLI.exe is missing.
- **UI-230** Fixed: Now using OnComponentDestroyed instead of FinishDestroy to unregister Wwise Game Objects, which had caused some Game Objects to be registered for too long.
- **UI-233** Fixed: Added validity checks on actors in the GetGameObjectID function.
- **UI-234** Fixed: Multiple viewports in the SetListener function are properly handled. This fixes a crash in AkAudioDevice.cpp.
- **UI-236** Fixed: The Location Type given to the GetAkComponent Blueprint node is properly handled.
- **UI-239** Fixed: No crash at engine shutdown as a result of the Bank Manager unloading SoundBanks that might already have been destroyed.
- Added a "Follow" check box to the AnimNotify_AkEvent. Leaving it unchecked will post the AkEvent at a specific location, instead of attaching to the parent.
- Added a new AnimNotify: AnimNotify_AkEventByName, allowing to post events using their name as a string.
- Added Debugging tools to Blueprints. For more information, see Debug Blueprint Functions
Unreal Engine 4.7 - Wwise v2014.1.3

- Added support for the Android platform.
- Now including a new demonstration game, based on Epic's First Person Template. For more information, see Sample Game.
- All Wwise Integration settings now have their page in the Unreal Project Settings window. No need to manually edit INI files anymore!
- **UI-157** Fixed: When generating the SoundBanks, add the Max Attenuation information to AkEvents. It can be retrieved via Blueprints.
- **UI-187** Fixed: Allow the preview of AkEvents from the Content Browser by pressing the space bar.
- **UI-188** Fixed: When an AkAmbientSound is about to be destroyed, cancel its event callback to avoid using a destroyed object in the callback.
- **UI-189** Fixed: Added an attenuation scaling factor to AkComponent, allowing to make each actor's attenuation radius unique.
- **UI-190** Fixed: Added a new Blueprint node that allows setting the exact list of loaded SoundBanks. For more information, see Load Banks
- **UI-193** Fixed: Get the UAkAudioEventFactory directly when creating UAkAudioEvents by Drag & Drop.
- **UI-194** Fixed: Optimized the finding of AkReverbVolumes at a location.
- **UI-195** Fixed: Added blueprint_actor_posteventbyname, features_blueprintcsoundbanks_loadbyname, and Unload Bank Blueprint nodes.
- **UI-196** Fixed: Added a configuration parameter for the maximum number of concurrent Reverb Volumes applied on a sound.
- **UI-201** Fixed: Added an UAkAudioEvent to AkComponent, to bring it in sync with Unreal's AudioComponent. An AkComponent can by added in a Blueprint via drag & drop of a UAkAudioEvent. An actor with an attached AkComponent will also show its attenuation radius in the Editor.
- **UI-204** Fixed: Added code to unload auto-loaded SoundBanks.
- **UI-205** Fixed: Don't post events in a world that does not allow audio playback.
- **UI-208** Fixed: Fixed a potential crash when generating SoundBanks.
Unreal Engine 4.6 - Wwise v2014.1.1

- **UI-183** Fixed: Use the debug SoundEngine libraries when UnrealBuildTool is configured to use the debug CRT libraries (bDebugBuildsActuallyUseDebugCRT = true).
- **UI-184** Fixed: Added a billboard component to AkComponent to make them visible in the Editor.
- **UI-186** Fixed: Allow drag & drop of .bnk files in the Content Browser to create UAkAudioBank assets.
Unreal Engine 4.5 - Wwise v2014.1

- **UI-172** Fixed: Resolved assert created by the AkReverbVolumeChannel collision channel.
- **UI-174** Fixed: Optimized the "Generate SoundBanks" window.
- **UI-175** Fixed: Added preprocessor guards to remove compilation errors on platforms not supported by this integration.
August 2014 - Wwise v2014.1

- **UI-166** Fixed: Auxiliary Bus properties are now applied on Ambient Sounds contained within Reverb Volumes.
- Updated Wwise version to 2014.1. If you wish to keep on using 2013.2.x, see Frequently Asked Questions.
August 2014 - Wwise v2013.2.9

- Added Japanese documentation.
July 2014 - Wwise v2013.2.9

- Added a SwitchDemo map to the UnrealWwiseDemo sample game, demonstrating the use of switches.
- **UI-161** Fixed: Removed the duplicate AkComponent created when starting an ambient sound. This prevented the Stop Ambient Sound method from working.
June 2014 - Wwise v2013.2.8

- **UI-152** Fixed: Add SoundBanks and streamed files to the packaging process.
April 2014 - Wwise v2013.2.7

- **UI-149** Fixed: Set proper default values for AkReverbVolumes.
March 2014 - Wwise v2013.2.6

- Visual Studio 2013 is now used by default. Please refer to the Requirements page for more details.
- **UI-146** Fixed: Fixed non-unity build compilation errors.
- **UI-147** Fixed:
  - Verify the SoundEngine is initialized in the AkComponent tick function;
  - Properly set the collision channel name for AkReverbVolumes;
  - Avoid loading banks in a commandlet;
  - Removed the automatic bank creation mechanism.
- **UI-148** Fixed: Use "Get Player Controller" in the RTPCDemo Blueprint to ensure sound keeps playing when transitioning between "Simulate" and "Possess" modes in Editor.
January 2014 - Wwise v2013.2.5

- Added an example of AkAnimNotify in ShooterGame. It is located in the FPP_RifleReload animation.
- **UI-131** Fixed: Spatialized sounds are now audible in the Animation Editor: Added a second listener for the Animation Editor window, and routed game objects created in this window to the new listener.

  **Warning:**
  This fix changes the AnimNotify_AkEvent Blueprint. If you have made modifications to it, be sure to keep a backup before merging this integration.

- **UI-134** Fixed: Removed the input flag StopWhenOwnerDestroyed to GetAkComponent. It was unused and could create confusion. Added a SetStopWhenOwnerDestroyed method to the AkComponent.
- **UI-136** Fixed: Allow Wwise Authoring to communicate with an Xbox One application.
- **UI-137** Fixed: Make sure a temporary game object (created by "Post Event at Location") is subject to AAkReverbVolumes.
- **UI-138** Fixed: Prevent the Sound Engine from creating UE errors while running in a commandlet. This is a workaround. Maybe the Sound Engine should not be initialized at all.
- **UI-139** Fixed: Removed FMath::Abs on Z projection when setting listener position in SoundEngine (Unreal code has been left untouched). This allows upside down listeners.
- **UI-145** Fixed: Fixed an error in the AkComponent::SetRTPCValue method that prevented RTPCs from being properly applied.
December 2013 - Wwise v2013.2.4

- Unreal Wwise integration now officially supports PS4 and Xbox One platforms!
- Added platform selection to bank generation dialog box.
- Added bank generation dialog box to build menu.
- LinkedProject configuration parameter has been moved from Engine/Config/BaseEditor.ini to <Your Game>/Config/DefaultGame.ini.
- Added simple occlusion support.
- Fixed a crash when getting the UAkComponent from another component which has no owner.
- **UI-130** Fixed: Fixed performance issues with the UAkComponent::UpdateAkReverbVolumeList method.
- **UI-130** Fixed: Added a flag to the UAkComponent specifying if it is influenced by reverb volumes.
- **UI-132** Fixed: Prevent UAkComponents from ticking on server configurations.
- **UI-135** Fixed: Expose RTPC interpolation time.
October 2013 - Wwise v2013.2.1

- Added bank generation for PlayStation4 and Xbox One.
- Added AkReverbVolume, allowing for mapping a volume in space to a Wwise auxiliary bus.
- **UI-128** Fixed: Implemented asynchronous IO access.
September 2013 - Wwise v2013.2.1

- Fixed UEngine::UseSound() returning false when only AkAudioDevice is present.
- Fixed crash in UAkComponent::PostAkEvent with null event.
- Fixed game object position with static actor.
August 2013 - Wwise v2013.2

- Fixed confusion between Actor, AkComponent, the Wwise game object and their respective lifespans:
  - AkComponent follows attachment semantics of Epic's AudioComponent. AkComponents are re-used when all attachment parameters match.
  - AkGameplayStatics contains global helpers targetting Actors for easy use in blueprints.
- Fixed missing Events and Banks in SoundBank Definition File.
- Fixed game object names in Wwise Profiler.
- Generating SoundBanks now triggers a refresh of loaded banks.
- Set RTPC Value global helper now supports setting a Game Parameter at global scope (no Actor target).
July 2013 - Wwise v2013.1.1

- Added message box in case of error during SoundBank Generation.
- New, distinct colors for Audiokinetic Event and Audiokinetic Bank asset types in the Content Browser.
- Fixed compilation on Xbox One and PS4 platforms.
- Fixed issue with AnimNotify_AkEvent using the wrong type.
- Fixed update of AkComponent Game Object position.
June 2013 - Wwise v2013.1.1

- Added DebugGame build configuration support.
- Partial Wwise SDK now included to help setting up the integration.
- Fixed xaudio2/mmdev DLL unload issues in Windows 64-bit build.
- Stopped matinee movie from playing in demomap - was preventing user input from working.
- Fixed AkAudioDevice listener position update in FLevelEditorViewportClient::UpdateAudioListener.
- Changed the AudioDeviceModuleName default value to XAudio2 on the windows platform.
May 2013 - Wwise v2013.1.1

- Introduced doxygen documentation.
- Now building using Wwise 2013.1.1 build 4677.
March 2013 - Wwise v2013.1

- Now building using Wwise 2013.1 beta build 4609.
Migration Notes

Migrating to the UE4.17/4.18 Wwise 2017.2.0 integration

A coordinate system conversion has been removed from the 2017.2.0 integration. This conversion was put in place to convert from the left-hand axis system used in Unreal Engine 4 (with the X and Y axes being the floor) to the default left-hand system used in Wwise Game Object 3D Viewer (with the X and Z axes being the floor). In order for the Wwise Game Object 3D Viewer to properly display Game Objects from Unreal, please go to the Game Object 3D Viewer settings and set the Floor Plane to X-Y:
Migrating Ak Acoustic Portal objects

Please note that in Wwise 2017.2.0, Spatial Audio Portals have a distinct orientation. The Portal's local Y-axis is the axis along which two adjacent Rooms are linked. It will be necessary to examine all existing Portals, and rotate (by 90 degrees, along the Z-axis) those that are not properly aligned in order to link Rooms along the Y-axis. A new visualization component has been added to the Portals to make the orientation of the Portal immediately apparent to the user. Portals that are incorrectly aligned may behave unexpectedly or return errors when sent to Wwise Spatial Audio.
Migrating to the UE4.11/4.12 Wwise 2016.1.1 integration

As part of the fix for WG-30404, the `AttenuationScalingFactor UPROPERTY` of `AkComponent` has been made read-only in Blueprints. To set its value, you now need to call the `SetAttenuationScalingFactor` method on `AkComponent`. 
Migrating from the UE4 Wwise integration source code to the plug-in version

Starting from Unreal Engine 4.11, the Wwise UE4 integration will now be distributed as a plug-in.

**Note:** Events can no longer be dragged from Wwise and dropped in the Unreal Content Browser to create corresponding AkAudioEvent objects. They can now be dragged directly from the Wwise Picker into the Unreal Content Browser to create their corresponding objects.

To update a UE4 project from UE 4.10 (or older) to UE 4.11 (or newer):

1. Back up your game project.
2. Install Unreal Engine from the Epic Games Launcher or compile your own Unreal Engine from source code.
3. Install the Wwise Unreal plug-in as an Engine or a Game plug-in, as outlined in Installation. Make sure that you use the SDK files you built when doing so.
4. Load your game project in the Unreal Editor.
5. If you installed the Wwise Unreal plug-in as an Engine plug-in, enable the Wwise plug-in by going to Edit > Plugins > Audio and selecting the "enable" option under the Wwise Unreal Engine 4 Integration section. You will need to reload your project to enable the plug-in.
6. If your project uses the AkEvent or the AkEventByName AnimNotify, you will be prompted to reload your project as some assets were modified to support the new plug-in model.

The last step is necessary because the AnimNotifies provided with the plug-in had to be moved from the Engine content folder to the plug-in's own content folder. Moving the references requires a project reload.

Once all the steps have been completed, you are ready to use the Wwise Unreal plug-in.
Migrating "...by name" methods

Starting with the 4.11 version of the Wwise Unreal plug-in, all "...by Name" methods (for example, PostEventByName or LoadBankByName) are deprecated. It is still possible to use strings to post Events or load SoundBanks by using the advanced fields in the Blueprint nodes. For example, if you wish to post an Event by name, you should now use the "Event Name" field of the "PostEvent" Blueprint node:
Migrating the Wwise project path

Starting with the 4.11 version of the Wwise Unreal plug-in, the Wwise project path, found in the Wwise settings, is now relative to the game folder, instead of the UE4Editor.exe file. The change should be automatically done the first time you run the 4.11 Wwise Unreal plug-in.
Known Issues

- **WG-29964** It is not possible to connect the Wwise Profiler to a Linux game.
- **WG-29968** Freeze on iOS platform. Make sure to disable the Unreal Engine audio device. See [Frequently Asked Questions](#) for more information.
- **WG-29981** Support for multiple independent Viewports.
- **WG-29982** Use Reverb Volumes flag doesn't work in real time: The Blueprint box "Set use reverb volumes" does not work as expected.
- **WG-30900** Add support for listener positioning within Animation editor.
- **WG-33970** Crash when deploying a packaged game using PAK files on Mac or iOS.